

Hugues Mounier

Differential flatness for
Neuroscience
population
dynamics

A preliminary study

Version 1 – November 2016

Hugues Mounier
Laboratoire des Signaux et Systèmes
CentraleSupélec
3, rue Joliot Curie
91192 GIF sur YVETTE
e-mail: hugues.mounier@l2s.centralesupelec.fr

CONTENTS

I	NEURAL MASS MODELS	3
1	SIMPLE AND COMMON NEURAL MASS MODELS	5
1.1	Scalar integrate and fire models	5
1.2	Two variables integrate and fire models	6
1.3	Two variables integrate and fire Izhikevich's models	6
1.4	Vectorial integrate and fire models	6
1.5	Neural mass Wilson-Cowan E-I networks	7
2	DIFFERENTIAL FLATNESS	9
2.1	Differential flatness notion	9
2.1.1	Dynamics and observation equations	9
2.1.2	Differential flatness definition	9
2.1.3	Parametrization	10
2.1.4	A word of methodology	11
2.2	Differential flatness of simple neural mass models	11
2.2.1	Weakly coupled E-I networks	11
2.2.2	Differential flatness of a weakly coupled E-I network	12
2.2.3	Differential flatness of Wilson Cowan's E-I network	12
2.2.4	Differential flatness of asymmetric Wilson Cowan's E-I network	12
3	DIFF. FLATNESS APPLICATIONS & EXTENSIONS	15
3.1	Differential flatness applications	15
3.1.1	Trajectory tracking	15
3.1.2	Feedforward to feedback switching	18
3.1.3	Cyclic character	18
3.1.4	Positivity & Boundedness	19
3.1.5	Simultaneous synchronisation & tracking	20
3.2	Flatness appls. for neural mass E-I networks	20
3.2.1	Trajectory tracking for weakly coupled E-I networks	20
3.2.2	Trajectory tracking for asymmetric Wilson-Cowan's E-I networks	22
3.2.3	Feedforward to feedback switching	23
3.3	Diff. flatness appls. for simplistic motor control	24
3.3.1	Two link arm model	24
3.3.2	Differential flatness and open loop control of the two link arm	25
3.3.3	End effector dynamics	26
3.3.4	Trajectory tracking of the two link arm	27
3.4	Extensions of differential flatness	30

3.5	Flatness of other neural mass models	31
3.5.1	Jansen and Rit model	31
II	NEURAL FIELD POPULATION MODELS	37
3.5.2	General case model	39
3.5.3	Parameters and variables assumptions	39
3.5.4	Neuron interaction strength examples	40
3.5.5	Simplification hypotheses	40
3.5.6	Pointwise neuron interaction strength models	41
3.5.7	Exponential type neuron interaction strength	42
3.6	A Jansen and Rit Neural field model	43
	BIBLIOGRAPHY	46
III	APPENDIX	51
A	NOTATIONS, TRANSFORMS AND SIGMOID FUNCTIONS	53
A.1	Functions and distributions	53
A.2	Transforms	54
A.3	Sigmoid functions	54
B	SOME FLATNESS SIMPLE CRITERIA	57
B.1	Necessary and sufficient conditions in peculiar cases	57
B.2	A necessary condition	58
B.3	Static state feedback linearizability criterion	58
B.3.1	Brief recall of differential geometry notions	58
C	DEFINITIONS FOR EXTENSIONS OF DIFFERENTIAL FLATNESS	61
C.1	Systems, dynamics and differential flatness	61
C.1.1	Basic definitions from differential algebra	61
C.1.2	Algebraic and transcendental extensions	61
C.1.3	Nonlinear systems and flatness	62
C.2	H-fields, Liouvillian and existential closedness	62
D	TWO LINK ARM INVERSE KINEMATICS	65
E	TWO LINK ARM CODE LISTING	67
E.1	Flatness based control of the arm	67

NOTATIONS

Some recurrent notations, transforms and sigmoid functions we shall be using throughout this document are given in Appendix A p. 53.

INTRODUCTION

The various objectives one wishes to attain through a controlled dynamical system almost always boil down to a system's behavior modification. One has at his disposal so called control variables whose aim is to steer the system. The behavior modification generally consist in tracking a prescribed trajectory, with stability. Note that the first phase (open loop trajectory tracking) is a feedforward one, while the second (with stability) is a feedback one. A possible methodology for controlling a system is then decomposed in two steps:

1. A so called open loop trajectory tracking, supposing the model perfect and the initial conditions perfectly known.
2. A feedback stabilizing the system around the reference trajectories, to compensate for model mismatch, poorly known initial conditions and external perturbations.

Simple and natural solutions to problem 1 are obtained through the differential flatness property, a notion due to Michel Fliess, Jean Lévine, Philippe Martin and Pierre Rouchon (Fliess et al., 1995). This property amounts to a parametrization of a dynamical system in terms of a so-called flat output: any variable in the system can be expressed through a function of the flat output components and a finite number of its derivatives. This parametrization yields expressions of all the system's variables without having to integrate any differential equation, which ensures fast computations. The control is in particular given through the flat output and its derivatives ; an inversion of the system control input to flat output is thus performed, without any integration. This notion has been extended to infinite dimensional systems, governed by partial differential equations (Woittennek and Mounier, 2010).

The present document is devoted to structural properties of neural population dynamics and especially their differential flatness. Several applications of differential flatness in the present context can be envisioned, among which: trajectory tracking, feedforward to feedback switching, cyclic character, positivity and boundedness.

Part I

NEURAL MASS MODELS

1

SIMPLE AND COMMON NEURAL MASS MODELS

The following models are quite simple models for neural populations with lumped space parameters (see, e.g. (Dayan and Abbott, 2005), Chapter 7, (Ermentrout and Terman, 2010), Chapter 11). The case of distributed parameter models, so-called neural field models, is considered in Section 3.5.2, p. 39.

1.1 SCALAR INTEGRATE AND FIRE MODELS

These type of models are of the form:

$$C\dot{v} = -g_L(v - v_L) + F(v) + I \quad (1.1)$$

where v is the membrane potential, determined with respect to the resting potential of the cell, τ_m is the membrane time constant, $F(v)$ is a spike generating current, and I is the total current elicited by synaptic inputs to the neuron.

Common types of models, each associated with a specific type of spike generating currents, are:

- The leaky integrate and fire, corresponding to $F = 0$
- The quadratic integrate and fire (or theta neuron), corresponding to

$$F(v) = \frac{g_L}{2\Delta_T} (v - v_T)^2 + g_L(v - v_T) - I_T$$

- The exponential integrate and fire, corresponding to

$$F(v) = g_L \Delta_T e^{\frac{v - v_T}{\Delta_T}}$$

1.2 TWO VARIABLES INTEGRATE AND FIRE MODELS

More general models adds a second variable coupled to the voltage (see (Izhikevich, 2010))

$$\begin{aligned}\dot{v} &= F(v) - \mu + I \\ \dot{\mu} &= a(bv - \mu)\end{aligned}$$

The function $F(v)$ describes the current–voltage characteristic of the membrane potential near the threshold, and it typically looks like a parabola (Izhikevich, 2003, 2004): $F(v) = v^2$. Other choices possible are

$$F(v) = |v|^3, \quad F(v) = \frac{1}{1-v}, \quad F(v) = |v|_+^n - v$$

An exponential spike generating current has been considered in (Brette and Gerstner, 2005) leading to the so-called adaptive exponential integrate and fire: $F(v) = e^v - v$, and (Touboul, 2009) suggested the quartic model $F(v) = v^4 + 2av$.

1.3 TWO VARIABLES INTEGRATE AND FIRE IZHIKEVICH'S MODELS

Another class of models is found in (Izhikevich, 2010):

$$\begin{aligned}\dot{v} &= F(v) - \mu(E - v) + I \\ \dot{\mu} &= a(bv - \mu)\end{aligned}$$

where v plays the role of a conductance and E is its reverse potential, which could be assumed to take values ± 1 or 0 after appropriate rescaling.

1.4 VECTORIAL INTEGRATE AND FIRE MODELS

Two types of integrate and fire models can be derived (see, e.g. (Ermentrout and Terman, 2010), Chapter 11):

$$\tau_m \dot{\mathbf{v}} = -\mathbf{v} + W\mathbf{F}(\mathbf{v}) + \tilde{\mathbf{I}} \quad (1.2)$$

where τ_m is the membrane time constant, and

$$\tau_d \dot{\boldsymbol{\rho}} = -\boldsymbol{\rho} + \mathbf{F}(W\boldsymbol{\rho} + \mathbf{I}) \quad (1.3)$$

where τ_d is the synaptic decay time.

- Remarks 1**
1. The choice of one of the models is based on time scale considerations (see, e.g. (Ermentrout and Terman, 2010), p. 335), where in (1.3) the temporal dynamics is dominated by the synaptic decay and in (1.3), the membrane time constant of the postsynaptic cell are small compared with the decay of the synapse.
 2. Note that the above two models can be shown to be equivalent (when $\tau_d = \tau_m = \tau$) in the following sense (see (Miller and Fumarola, 2012)). If ρ is a solution of the membrane model (1.3), then $W\rho + I$ is a solution of (1.2). Indeed, setting $v = W\rho + I$, one obtains

$$\begin{aligned}\tau\dot{v} &= \tau W\dot{\rho} + \tau\dot{I} = W(-\rho + F(W\rho + I)) + \tau\dot{I} \\ &= -(v - I) + WF(v) + \tau\dot{I} \\ &= -v + WF(v) + \tilde{I}\end{aligned}$$

1.5 NEURAL MASS WILSON-COWAN E-I NETWORKS

Consider the simplest form of network, a pair of mutually coupled local populations of excitatory and inhibitory neurons, also called E-I network (see, e.g. (Bressloff, 2014), Subsection 6.2, p. 238). This model was originally developed by Wilson and Cowan (see, e.g. (Ermentrout and Terman, 2010), Subsection 11.3, p. 344), and has the form

$$\tau_e \dot{v}_e = -v_e + F_e(w_{ee} v_e - w_{ie} v_i + I_e) \quad (1.4a)$$

$$\tau_i \dot{v}_i = -v_i + F_i(w_{ii} v_i - w_{ei} v_e + I_i) \quad (1.4b)$$

where v_e and v_i are the proportion of excitatory and inhibitory cells firing per unit time, the activations are nonlinear functions (typically sigmoidal) F_e, F_i of the presently active proportion of cells, w_* are the strength of the connections.

The matrix form of the previous model is

$$\tau_e \dot{v}_e = -v_e + F_e(W_{ee} v_e - W_{ie} v_i + I_e) \quad (1.5a)$$

$$\tau_i \dot{v}_i = -v_i + F_i(W_{ii} v_i - W_{ei} v_e + I_i) \quad (1.5b)$$

2

DIFFERENTIAL FLATNESS

2.1 DIFFERENTIAL FLATNESS NOTION

2.1.1 Dynamics and observation equations

Consider a system given by the dynamics equation and the observation equation

$$\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u}) \quad \text{dynamics equation} \quad (2.1a)$$

$$\mathbf{y}_m = h(\mathbf{x}) \quad \text{observation equation} \quad (2.1b)$$

with $\mathbf{x}(t) = (x_1(t), \dots, x_n(t))$, the *state*, or, in Karl Friston's terms the *hidden variables* (see, e.g. (Friston, 2012)), i.e. the controlled variables, $\mathbf{u}(t) = (u_1(t), \dots, u_m(t))$, the *control input*, functions enabling an action on the process (typically input current), and $\mathbf{y}_m = (y_{m1}(t), \dots, y_{mp}(t))$ the *output*, measured functions enabling to sense the environment (quantities coming from sensors).

Note that the dynamics equations form an *undetermined* system of differential equations, since the control functions $\mathbf{u}(t)$ are not a priori determined. Once the control variables are fixed (i.e. substituted with known functions of time), the system (2.1) becomes determined (i.e. can be integrated). The *state* variables represent the instantaneous memory of the system: once the control variables have been determined, the knowledge of the state variables (at time t) enables to predict the future state (at time $t + dt$).

Another formulation is the following: the state of a dynamical system is a set of physical quantities the specification of which (in the absence of external excitation) completely determines the evolution of the system.

2.1.2 Differential flatness definition

The notion of differential flatness (see (Fliess et al., 1995)) is a form of controllability for non linear dynamical systems which is espe-

cially well suited for trajectory tracking problems. It amounts to a parametrization of the system without integration of any differential equation. Although the mathematical property seems quite strong, it appears that this notion is commonly encountered in practice (see, e.g. (Rouchon, 2001, Martin and Rouchon, 2008) for a catalog of differentially flat systems). We shall give below a definition for such systems and illustrate this through simple examples derived from the well known Wilson and Cowan's model. Some more details about this property is given in the appendices.

Definition 1 *The system*

$$\dot{x} = f(x, u) \quad (2.2)$$

with $x(t) \in \mathbb{R}^n$ and $u(t) \in \mathbb{R}^m$ is differentially flat if there exists a set of variables, called a flat output,

$$y = h(x, u, \dot{u}, \dots, u^{(r)}), \quad y(t) \in \mathbb{R}^m, r \in \mathbb{N} \quad (2.3)$$

such that

$$x = A(y, \dot{y}, \dots, y^{(\rho_x)}) \quad (2.4a)$$

$$u = B(y, \dot{y}, \dots, y^{(\rho_u)}) \quad (2.4b)$$

with q an integer, and such that the system equations

$$\frac{dA}{dt}(y, \dots, y^{(q+1)}) = f(A(y, \dots, y^{(q)}), B(y, \dots, y^{(q+1)}))$$

are identically satisfied.

2.1.3 Parametrization

For any flat output given through a function of the form $t \in \mathbb{R} \rightarrow y(t)$, the trajectory of the system $x(t), u(t)$ are given by:

$$x(t) = A(y(t), \dot{y}(t), \dots, y^{(\rho_x)}(t)) \quad (2.5a)$$

$$u(t) = B(y(t), \dot{y}(t), \dots, y^{(\rho_u)}(t)) \quad (2.5b)$$

There is a one to one correspondance between the system trajectories and the ones given by the flat output.

2.1.4 A word of methodology

The preceding notion will be used to obtain so called “open loop” controls, that is control laws which will ensure the tracking of the reference flat outputs when the *model is assumed to be perfect* and *the state initial conditions are assumed to be exactly known*. Since this is never the case in practice, one needs some feedback schemes that will ensure asymptotic convergence to zero of the tracking errors. Our framework can thus be decomposed in two steps:

1. Design of the reference trajectory of the flat outputs; off-line computation of the open loop controls.
2. Inline computation of the complementary closed loop controls in order to stabilize the system around the reference trajectories.

Why is this two step design better suited than a classical stabilization scheme? The first step obtains a first order solution to the tracking problem, while *following the model* instead of forcing it (like in a usual pure stabilization scheme). The second step is a refinement one, and the error between the actual values and the tracked references will be much smaller than in the pure stabilization case.

2.2 DIFFERENTIAL FLATNESS OF SIMPLE NEURAL MASS MODELS

2.2.1 Weakly coupled E-I networks

Consider a Wilson-Cowan model where $w_{ie} \ll 1$. Hence (1.4a) reduces to

$$\tau_e \dot{v}_e = -v_e + F_e(w_{ee} v_e + I_e)$$

Alternatively, one could also consider the other limit case where $w_{ei} \ll 1$ where (1.4b) reduces to

$$\tau_i \dot{v}_i = -v_i + F_i(w_{ii} v_i + I_i)$$

Whatever case we consider, we shall abbreviate it by the following highly simplified model

$$\tau \dot{v} = -v + F(w v + I) \tag{2.6}$$

where the subscript has been dropped for convenience. This model, although simplistic, is considered here because of its simplicity for pedagogical purposes. Set

$$\phi = F^{-1}$$

where F is a sigmoid function (see A.3, p. 54).

2.2.2 Differential flatness of a weakly coupled E-I network

The model depicted by (2.6) is differentially flat, with v as a flat output. Indeed, one has

$$wv + I = \phi(\tau\dot{v} + v)$$

and the input I is given by

$$I = -wv + \phi(\tau\dot{v} + v) \quad (2.7)$$

2.2.3 Differential flatness of Wilson Cowan's E-I network

The E-I network equations (1.4a)–(1.4b)

$$\begin{aligned} \tau_e \dot{v}_e &= -v_e + F_e(w_{ee} v_e - w_{ie} v_i + I_e) \\ \tau_i \dot{v}_i &= -v_i + F_i(w_{ii} v_i - w_{ei} v_e + I_i) \end{aligned}$$

rewrite

$$\begin{aligned} w_{ee} v_e - w_{ie} v_i + I_e &= F_e^{-1}(\tau_e \dot{v}_e + v_e) \\ w_{ii} v_i - w_{ei} v_e + I_i &= F_i^{-1}(\tau_i \dot{v}_i + v_i) \end{aligned}$$

This model is thus differentially flat with flat output (v_e, v_i) :

$$\begin{aligned} I_e &= w_{ie} v_i - w_{ee} v_e + F_e^{-1}(\tau_e \dot{v}_e + v_e) \\ I_i &= -w_{ei} v_e + w_{ii} v_i + F_i^{-1}(\tau_i \dot{v}_i + v_i) \end{aligned}$$

2.2.4 Differential flatness of asymmetric Wilson Cowan's E-I network

Consider the E-I Wilson-Cowan equations (1.4a)–(1.4b) with statically coupled external currents

$$I_e = (1 + a)I, \quad I_i = (1 - a)I$$

with $a \in [-1, 1]$ an asymetry factor. The model (1.4a)–(1.4b) then becomes (see, e.g. (Ermentrout and Terman, 2010), Section 11.3, p. 349)

$$\tau_e \dot{v}_e = -v_e + F_e((1+a)I - w_{ie} v_i)$$

$$\tau_i \dot{v}_i = -v_i + F_i((1-a)I - w_{ei} v_e)$$

In the asymmetric case, i.e. $a = -1$ the preceding equations are

$$\tau_e \dot{v}_e = -v_e + F_e(-w_i v_i) \tag{2.8a}$$

$$\tau_i \dot{v}_i = -v_i + F_i(2I - w_e v_e) \tag{2.8b}$$

Then, v_e is a flat output. Indeed, one gets

$$v_i = -\frac{1}{w_i} F_e^{-1}(\tau_e \dot{v}_e + v_e)$$

$$I = \frac{1}{2} \left[w_e v_e + F_i^{-1}(\tau_i \dot{v}_i + v_i) \right]$$

3

DIFFERENTIAL FLATNESS APPLICATIONS AND EXTENSIONS

3.1 DIFFERENTIAL FLATNESS APPLICATIONS

A number of applications of differential flatness can be envisioned, among which:

- *Trajectory tracking.*
- *Feedforward to feedback switching.*
- *Cyclic character.*
- *Positivity & boundedness.*
- *Simultaneous synchronisation & tracking.*

3.1.1 Trajectory tracking

Flatness and feedback linearization

A characterization of flat systems that appears very useful for stabilized trajectory tracking is the following

Proposition 1 *A system is flat if, and only if, it is linearizable by endogenous feedback and change of coordinates.*

A dynamic feedback is called endogenous if it does not include any external dynamics. More precisely

Definition 2 *Consider the dynamics $\dot{x} = f(x, u)$. The feedback*

$$u = \tilde{\zeta}(x, z, v)$$

$$\dot{z} = \zeta(x, z, v)$$

(where v is the new input) is called a dynamic endogenous feedback if the original dynamics $\dot{x} = f(x, u)$ is equivalent to the transformed one

$$\begin{aligned}\dot{x} &= f(x, \zeta(x, z, v)) \\ \dot{z} &= \zeta(x, z, v)\end{aligned}$$

Two systems are called equivalent if there exists an invertible transformation which exchanges their trajectories.

A more restrictive notion is the one of static state feedback, as described below.

Definition 3 Consider the dynamics $\dot{x} = f(x, u)$. The feedback

$$u = \zeta(x, v)$$

(where v is the new input) is called a static feedback if the original dynamics $\dot{x} = f(x, u)$ is transformed to

$$\dot{x} = f(x, \zeta(x, v))$$

See the Subsection B.3, p. 58 for a static state feedback linearization criterion.

Dynamical extension algorithm

This procedure enables one to know if an m -uple (y_1, \dots, y_m) is a flat output or not. Meanwhile, we shall obtain a linearizing feedback.

PHASE I – Gathering the so called weak brunovsky indices.

- 1) Differentiate y_1 until a combination of controls appears. Note κ_1 the number of successive differentiations $y_1^{(\kappa_1)} = f_1$
- 2) Differentiate y_2 until a combination of controls (independent of the previous ones) appears. Note κ_2 the number of successive differentiations $y_2^{(\kappa_2)} = f_2$
- \vdots
- m) Differentiate y_m until a combination of controls (independent of the previous ones) appears. Note κ_m the number of successive differentiations $y_m^{(\kappa_m)} = f_m$

PHASE II – Deciding the flatness character.

Then, if $\kappa_1 + \dots + \kappa_m = n$ (n being the state dimension), the system admits (y_1, \dots, y_m) as a flat output. If not, (y_1, \dots, y_m) isn't a flat output.

PHASE III – Obtaining the linearizing feedback.

The linearizing feedback is given by $f_1 = v_1, \dots, f_m = v_m$.

Closed loop trajectory tracking

The open loop control laws suppose that the model is perfect and that the initial conditions are exactly known. Since this is never the case in practice, we add corrective terms to the open loop controls derived above in order to stabilize the system around the reference trajectories.

More precisely, considering a flat dynamics $\dot{x} = f(x, u)$, we want to derive a controller able to follow any reference trajectory $t \mapsto \mathbf{y}_r(t)$. In order to compensate for model mismatch and poorly known initial conditions, one has to complement the open loop (obtained through flatness) with a closed loop corrective term depending on the error $\mathbf{y}(t) - \mathbf{y}_r(t)$.

Knowing the dynamics is flat, with flat output \mathbf{y} , it can be transformed via endogenous feedback and coordinate change to a linear dynamics of the form

$$\begin{aligned} y_1^{(\kappa_1)} &= v_1 \\ &\vdots \\ y_m^{(\kappa_m)} &= v_m \end{aligned}$$

with the new input (v_1, \dots, v_m) . Then, the elementary tracking feedback

$$\begin{aligned} v_i &= y_{ir}^{(\kappa_i)} - \sum_{j=0}^{\kappa_i-1} k_{ij}(y_i^{(j)} - y_{ir}^{(j)}), \quad i = 1, \dots, m \\ &= y_{ir}^{(\kappa_i)} - \sum_{j=0}^{\kappa_i-1} k_{ij}e_i^{(j)} \end{aligned}$$

with appropriately chosen k_{ij} gains renders the error dynamics asymptotically stable:

$$e_i^{(\kappa_i)} = \sum_{j=0}^{\kappa_i-1} -k_{ij}e_i^{(j)}, \quad i = 1, \dots, m$$

3.1.2 Feedforward to feedback switching

Open and closed loop

The so-called *open loop* control u_o is obtained through (2.5b)

$$u(t) = B(y(t), \dot{y}(t), \dots, y^{(\rho_u)}(t))$$

by replacing y with a sufficiently differentiable trajectory $y_r(t)$:

$$u_o(t) = B(y_r(t), \dot{y}_r(t), \dots, y_r^{(\rho_u)}(t))$$

The use of this control law would lead to the desired tracking behavior $y = y_r$ if the model (2.1a) was perfect and if the initial conditions on y was precisely known. Since this is never the case in practice, one has to use closed loop feedback laws, such as the ones elaborated in the previous Subsection 3.1.1, p. 15. The difference between open and closed loop control laws can be bounded by the tracking error and its derivatives. The simple weakly coupled E-I network example is examined in Subsection 3.2.3, p. 23.

Temporal switching from feedforward to feedback

Consider the following control law

$$u(t) = (1 - \sigma(t - t_{sw}))u_o(t) + \sigma(t - t_{sw})u_c(t) \quad (3.1)$$

with σ a sigmoid function, for example of the form

$$\sigma(t) = \frac{1}{1 + e^{\frac{-t-\beta}{\alpha}}}, \quad \sigma(t) = \frac{1 + \tanh(\alpha t)}{2}$$

Thus, from $t = 0$ to $t = t_{sw} - d$ for some $d > 0$, we have $u \approx u_o$, and from $t = t_{sw} + d$, $u \approx u_c$. This is the kind of control human beings tend to adopt for example in gesture control. When grasping a glass of water, the first part of the gesture is done in open loop, quickly and inaccurately; the second part of it is done with visual feedback, much more slowly but precisely.

3.1.3 Cyclic character

When the flat output is cyclic, i.e.

$$(1 - \Delta_\tau)y(t) = y(t) - y(t - \tau) = 0$$

Then, all the variables which are expressed as functions of $y(t)$, that is all the variables when the system is flat, are also cyclic: for a variable z which is expressed as $z = C(y, \dot{y}, \dots, y^{(\eta)})$

$$\begin{aligned} (1 - \Delta_\tau)z &= (1 - \Delta_\tau)C(y, \dot{y}, \dots, y^{(\eta)}) \\ &= C((1 - \Delta_\tau)y, (1 - \Delta_\tau)\dot{y}, \dots, (1 - \Delta_\tau)y^{(\eta)}) \\ &= 0 \end{aligned} \tag{3.2}$$

More generally, if the flat output satisfies a difference equation:

$$p(\Delta_\tau)y = 0$$

where p is a polynomial, then any variable of the flat system with flat output y also satisfies the same difference equation.

3.1.4 Positivity & Boundedness

The goal is here to specify the reference trajectory in order to enforce certain properties for various system variables. Two main cases can be considered: The one of cyclic reference trajectories and the one of non cyclic ones.

Cyclic reference trajectories

The flat output reference trajectories being cyclic can be expressed through a Fourier series

$$\forall i = 1, \dots, m, \quad y_i = \sum_{n=1}^{\infty} \tilde{\zeta}_{i,n} e^{\frac{2j\pi}{n}} \tag{3.3}$$

Since all variables are also cyclic (see (3.2)) they can also be expressed through a Fourier series

$$z = \sum_{n=1}^{\infty} \zeta_n e^{\frac{2j\pi}{n}} \tag{3.4}$$

One then has some relations expressing the ζ_n through the $\tilde{\zeta}_{i,n}$:

$$\zeta_n = \phi_n(\tilde{\zeta}_{1,n}, \dots, \tilde{\zeta}_{m,n})$$

And the positivity can be expressed through a sum of squares type formula (see, e.g. (Dumitrescu, 2007)). Several matlab packages are available for finding sum of squares decompositions of real multivariate polynomials; the most popular ones are SOSTOOLS (see <http://>

www.cds.caltech.edu/sostools/), YALMIP (see <http://users.isy.liu.se/johanl/yalmip/>, and especially <http://users.isy.liu.se/johanl/yalmip/pmwiki.php?n=Examples.MoreSOS>) and GloptiPoly (see <http://homepages.laas.fr/henrion/software/gloptipoly/>).

Non cyclic reference trajectories

One option is to take in the flat output $\mathbf{y} = (y_1, \dots, y_m)$ all the components y_i s as polynomial splines. If the firing rate function is taken to be of Naka Rushton type, then all inequalities will boil down to expressions of the form

$$P_i(\mathbf{y}, \dot{\mathbf{y}}, \dots, \mathbf{y}^{(\rho)}) > 0, \quad i = 1, \dots, m$$

where the P_i s are polynomials in their variables. Since the y_i s are polynomial splines, $P_i(\mathbf{y}, \dots, \mathbf{y}^{(\rho)})$ will be another polynomial spline. And any approximating polynomial spline is contained in the convex hull of its control points. One then can choose the lowest of these to be positive, to ensure the above inequality to be fulfilled.

3.1.5 Simultaneous synchronisation & tracking

One considers here two (or more generally N) oscillators coupled via their input:

$$\begin{aligned} \dot{x}_1 &= f_1(\mathbf{x}, u) \\ \dot{x}_2 &= f_2(\mathbf{x}, u) \end{aligned}$$

The flatness of this system ensures not only that synchronisation is possible, but also that any periodic trajectory (of the flat output) may be tracked, which is a much stronger result.

3.2 DIFFERENTIAL FLATNESS APPLICATIONS FOR SIMPLE NEURAL MASS E-I NETWORKS

3.2.1 Trajectory tracking for weakly coupled E-I networks

Recall the weakly coupled E-I network (2.6), p. 11:

$$\tau \dot{v} = -v + F(wv + I) \tag{3.5}$$

In a trajectory tracking, one chooses a reference trajectory ν_r and one wants that $\lim_{t \rightarrow \infty} \nu = \nu_r$, or, what is the same

$$\lim_{t \rightarrow \infty} e_\nu = 0, \quad \text{where} \quad e_\nu = \nu - \nu_r$$

This behavior can be enforced through the following desired error dynamics

$$\dot{e}_\nu = -\lambda e_\nu \quad (3.6)$$

where $\lambda > 0$ is a user chosen gain ruling the tracking error convergence speed. In order to obtain the desired behavior (3.6), one has to set in (2.6):

$$-\nu + F(w\nu + I_c) = -\tau\lambda e_\nu + \tau\dot{\nu}_r$$

where I_c is the closed loop control law. The preceding equation can be rewritten as

$$w\nu + I_c = \phi(\nu - \tau\lambda e_\nu + \tau\dot{\nu}_r)$$

which yields the following closed loop tracking feedback law

$$I_c = -w\nu + \phi(\tau\dot{\nu}_r + \nu - \tau\lambda e_\nu) \quad (3.7)$$

which ensures, through (3.6), the tracking of the reference trajectory ν_r for the system (2.6) with stability.

Remark 1 *The application of the preceding extension algorithm is quite trivial since the system is fairly simple:*

- *Gathering the so called weak brunovsky indices.*
The flat output ν is differentiated once in equation (2.6) where the control I is already present, hence $\kappa_1 = 1$.
- *Deciding the flatness character.*
Since the dimension of the state is $n = 1$, $\sum_i \kappa_i = \kappa_1 = n$ and the system is flat with flat output ν .
- *Obtaining the linearizing feedback.*
The linearizing feedback is given by:

$$\frac{1}{\tau} (-\nu + F(w\nu + I)) = \dot{\nu} \quad (3.8)$$

This feedback transforms the dynamics (2.6) into the following linear one:

$$\dot{v} = v$$

and the elementary tracking feedback is

$$v = \dot{v}_r - \lambda e_v \quad (3.9)$$

Thus, the original tracking feedback law is obtained from (3.8) and (3.9):

$$I = -w v + \phi(\tau \dot{v}_r + v - \tau \lambda e_v)$$

3.2.2 Trajectory tracking for asymmetric Wilson-Cowan's E-I networks

Recalling the equations of the asymmetric Wilson-Cowan's E-I network (2.8a)–(2.8b)

$$\begin{aligned} \tau_e \dot{v}_e &= -v_e + F_e(-w_i v_i) \\ \tau_i \dot{v}_i &= -v_i + F_i(2I - w_e v_e) \end{aligned}$$

and differentiating the first equation in v_e , we get the flat output dynamics

$$\begin{aligned} \tau_e \ddot{v}_e &= -\dot{v}_e - w_i F'_e(-w_i v_i) \dot{v}_i \\ &= -\dot{v}_e + \frac{w_i}{\tau_i} F'_e(-w_i v_i) (v_i - F_i(2I - w_e v_e)) \end{aligned} \quad (3.10)$$

The desired dynamics being

$$\ddot{e}_{er} = -\lambda_e e_{er} - \mu_e \dot{e}_{er}, \quad \text{where } e_{er} = v_e - v_{er}$$

the right hand side of (3.10) is then taken to be

$$\begin{aligned} \dot{v}_e + \frac{w_i}{\tau_i} F'_e(-w_i v_i) (-v_i + F_i(2I - w_e v_e)) = \\ \tau_e (\ddot{v}_{er} + \lambda_e e_{er} + \mu_e \dot{e}_{er}) \end{aligned}$$

Thus we get

$$\begin{aligned} -v_i + F_i(2I - w_e v_e) = \\ \frac{\tau_i \tau_e}{w_i F'_e(-w_i v_i)} \left(\frac{1}{\tau_e} \dot{v}_e + \ddot{v}_{er} + \lambda_e e_{er} + \mu_e \dot{e}_{er} \right) \end{aligned}$$

and the tracking control feedback loop is obtained as

$$I = \frac{1}{2} \left[w_e v_e + F_i^{-1} \left(v_i + \frac{\tau_i \tau_e}{w_i F'_e(-w_i v_i)} \left(\frac{1}{\tau_e} \dot{v}_e + \ddot{v}_{er} + \lambda_e e_{er} + \mu_e \dot{e}_{er} \right) \right) \right]$$

3.2.3 Feedforward to feedback switching

Open and closed loop

The so-called open loop control law is obtained when replacing v by the reference trajectory v_r in (2.7):

$$I_o = -w v_r + \phi(\tau \dot{v}_r + v_r) \quad (3.11)$$

The use of this control law would lead to the desired tracking behavior $v = v_r$ if the model (2.6) was perfect and if the initial conditions on v was precisely known.

This type of law is typically used by the brain, after training, for quick movements where the sensory system is bypassed. When the sensory system is used, the so-called closed loop control law (3.7) is applied.

The difference between I_c and I_o is

$$\begin{aligned} I_o - I_c &= w e_v - \phi(\tau \dot{v}_r + v - \tau \lambda e_v) + \phi(\tau \dot{v}_r + v_r) \\ &= w e_v - \phi(\tau \dot{v}_r + v_r + (1 - \tau \lambda) e_v) + \phi(\tau \dot{v}_r + v_r) \end{aligned}$$

Then, supposing ϕ to be globally γ -lipschitz:

$$|\phi(\tau \dot{v}_r + v_r + (1 - \tau \lambda) e_v) - \phi(\tau \dot{v}_r + v_r)| \leq \gamma |1 - \tau \lambda| |e_v|$$

Hence the difference $I_c - I_o$ admits the following bound

$$|I_c - I_o| \leq (\alpha + \gamma |1 - \tau \lambda|) |e_v|$$

Thus, if the tracking error is small, I_c is in a neighborhood of I_o .

Temporal switching from feedforward to feedback

Consider the following control law

$$I(t) = (1 - \sigma(t - t_{sw})) I_o(t) + \sigma(t - t_{sw}) I_c(t) \quad (3.12)$$

with σ a sigmoid function (see A.3, p. 54). Thus, from $t = 0$ to $t = t_{sw} - d$ for some $d > 0$, we have $I \approx I_o$, and from $t = t_{sw} + d$, $I \approx I_c$. This is the kind of control human beings tend to adopt for example in gesture control. When grasping a glass of water, the first part of the gesture is done in open loop, quickly and inaccurately; the

second part of it is done with visual feedback, much more slowly but precisely. The expression can alternatively be expressed as:

$$\begin{aligned} I &= (1 - \sigma_{sw})\phi(\tau\dot{v}_r + v_r) + \sigma_{sw}(-we_v + \phi(\tau\dot{v}_r + v - \tau\lambda e_v)) \\ &= \phi(\tau\dot{v}_r + v_r) + \sigma_{sw}(-we_v + \Delta\phi(v, v_r)) \end{aligned}$$

where $\sigma_{sw} = \sigma(t - t_{sw})$ and

$$\Delta\phi(v, v_r) = \phi(\tau\dot{v}_r + v - \tau\lambda e_v) - \phi(\tau\dot{v}_r + v_r)$$

3.3 DIFFERENTIAL FLATNESS APPLICATIONS FOR SIMPLISTIC MOTOR CONTROL

3.3.1 Two link arm model

Consider a two link robot arm acting as a simplistic model of a human arm:

$$M_{11}\theta_1 + M_{12}\theta_2 + C_1(\theta, \dot{\theta}) + G_1(\theta) = T_1 \quad (3.13a)$$

$$M_{21}\theta_1 + M_{22}\theta_2 + C_2(\theta, \dot{\theta}) + G_2(\theta) = T_2 \quad (3.13b)$$

where θ_1 is the angle of the first arm, θ_2 of the second, $\theta = (\theta_1, \theta_2)^T$, M_{ij} are equivalent masses, C_i are the coriolis forces, G_i are the gravity forces, and T_i are the control torques. The expressions for the C_i 's and the G_i 's are the following:

- o The inertia expressions are

$$M_{11} = J_1 + J_2 + m_1 r_1^2 + m_2 (l_1^2 + r_1^2 + 2l_1 r_2 \cos \theta_2) \quad (3.14a)$$

$$M_{12} = M_{21} = J_2 + m_2 (r_2^2 + l_1 r_2 \cos \theta_2) \quad (3.14b)$$

$$M_{22} = J_2 + m_2 r_2^2 \quad (3.14c)$$

where J_i is the inertia of link i , m_i its mass, l_i its length, and r_i the distance from the beginning of the link to its center of mass.

- o The Coriolis terms are given by:

$$C_1 = -m_2 l_1 \theta_2^2 r_2 \sin \theta_2 - 2m_2 l_1 \dot{\theta}_1 \dot{\theta}_2 r_2 \sin \theta_2 \quad (3.15a)$$

$$C_2 = m_2 l_1 \dot{\theta}_1^2 r_2 \sin \theta_2 \quad (3.15b)$$

- o And the gravity terms are

$$G_1 = (m_2 l_1 + m_1 r_1)g \sin \theta_1 + m_2 r_2 g \sin(\theta_1 + \theta_2) \quad (3.16a)$$

$$G_2 = m_2 r_2 g \sin(\theta_1 + \theta_2) \quad (3.16b)$$

Equations (3.13) can be rewritten in a vectorial form; to this purpose, set

$$M = \begin{pmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{pmatrix}, \quad \theta = \begin{pmatrix} \theta_1 \\ \theta_2 \end{pmatrix}$$

$$C = \begin{pmatrix} C_1 \\ C_2 \end{pmatrix}, \quad G = \begin{pmatrix} G_1 \\ G_2 \end{pmatrix}, \quad T = \begin{pmatrix} T_1 \\ T_2 \end{pmatrix}$$

Then, model (3.13) becomes

$$M\ddot{\theta} + C(\theta, \dot{\theta}) + G(\theta) = T \quad (3.17)$$

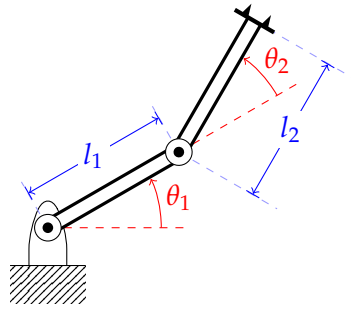


Figure 3.1: A two link robot arm.

3.3.2 Differential flatness and open loop control of the two link arm

Remark 2 Model (3.17) is differentially flat, with θ as a flat output. Indeed, the inputs T are directly expressed in terms of θ and its derivatives:

$$T = M\ddot{\theta} + C(\theta, \dot{\theta}) + G(\theta)$$

and the open loop control for a trajectory θ_{1r}, θ_{2r} given by

$$T_r = M\ddot{\theta}_r + C(\theta_r, \dot{\theta}_r) + G(\theta_r)$$

Knowing that the desired trajectory is generally not given in terms of θ_1, θ_2 but in terms of the end effector coordinates h_x, h_y , we have to express the former in terms of the latter. The end effector (e.g. the wrist) coordinates are given by:

$$h_x = l_1 \cos \theta_1 + l_2 \cos(\theta_1 + \theta_2) \quad (3.18a)$$

$$h_y = l_1 \sin \theta_1 + l_2 \sin(\theta_1 + \theta_2) \quad (3.18b)$$

The inversion of these formulae is detailed in Appendix D, p. 65. We shall here give the final expressions:

$$\theta_1 = \arctan\left(\frac{h_y}{h_x}\right) - \arctan\left(\frac{l_2 \sin \theta_2}{l_1 + l_2 \cos \theta_2}\right) \quad (3.19a)$$

$$\theta_2 = \arctan\left(\pm \frac{\sqrt{1 - \bar{h}^2}}{\bar{h}}\right) \quad (3.19b)$$

$$\bar{h} = \frac{h_x^2 + h_y^2 - l_1^2 - l_2^2}{2l_1 l_2}$$

3.3.3 End effector dynamics

The dynamcis in θ is given by:

$$\ddot{\theta} = -M^{-1}(C + G) + M^{-1}T \quad (3.20)$$

and the dynamcis in the end effector, i.e. in h_x, h_y is obtained through a double differentiation of (3.18). A first differentiation yields

$$\begin{aligned} \dot{h}_x &= -l_1 \dot{\theta}_1 \sin \theta_1 - l_2 (\dot{\theta}_1 + \dot{\theta}_2) \sin(\theta_1 + \theta_2) \\ \dot{h}_y &= l_1 \dot{\theta}_1 \cos \theta_1 + l_2 (\dot{\theta}_1 + \dot{\theta}_2) \cos(\theta_1 + \theta_2) \end{aligned}$$

And then

$$\ddot{h}_x = -h_y \ddot{\theta}_1 - l_2 \sin(\theta_1 + \theta_2) \ddot{\theta}_2 - \phi_x(\theta, \dot{\theta}) \quad (3.21a)$$

$$\ddot{h}_y = h_x \ddot{\theta}_1 + l_2 \cos(\theta_1 + \theta_2) \ddot{\theta}_2 - \phi_y(\theta, \dot{\theta}) \quad (3.21b)$$

with

$$\begin{aligned} \phi_x(\theta, \dot{\theta}) &= l_1 \dot{\theta}_1^2 \cos \theta_1 + l_2 (\dot{\theta}_1 + \dot{\theta}_2)^2 \cos(\theta_1 + \theta_2) \\ \phi_y(\theta, \dot{\theta}) &= l_1 \dot{\theta}_1^2 \sin \theta_1 + l_2 (\dot{\theta}_1 + \dot{\theta}_2)^2 \sin(\theta_1 + \theta_2) \end{aligned}$$

Thus, one gets

$$\begin{pmatrix} \ddot{h}_x \\ \ddot{h}_y \end{pmatrix} = \begin{pmatrix} -h_y & -l_2 \sin(\theta_1 + \theta_2) \\ h_x & l_2 \cos(\theta_1 + \theta_2) \end{pmatrix} \begin{pmatrix} \ddot{\theta}_1 \\ \ddot{\theta}_2 \end{pmatrix} - \begin{pmatrix} \phi_x \\ \phi_y \end{pmatrix}$$

Or, in other terms

$$\ddot{h} = H \ddot{\theta} - \phi$$

With the following notations

$$H = \begin{pmatrix} -h_y & -l_2 \sin(\theta_1 + \theta_2) \\ h_x & l_2 \cos(\theta_1 + \theta_2) \end{pmatrix}, \quad \mathbf{h} = \begin{pmatrix} h_x \\ h_y \end{pmatrix} \quad (3.22)$$

And, using (3.20), one gets the dynamics in the end effector \mathbf{h} :

$$\ddot{\mathbf{h}} = -H M^{-1}(\mathbf{C} + \mathbf{G} - \mathbf{T}) - \boldsymbol{\phi} \quad (3.23)$$

3.3.4 Trajectory tracking of the two link arm

The system (3.23) is differentially flat, with flat output h_x, h_y . Indeed equations (3.19) yield the expressions of θ_1 and θ_2 in terms of h_x, h_y , and \mathbf{T} is given by:

$$\mathbf{T} = \mathbf{C} + \mathbf{G} + M H^{-1}(\ddot{\mathbf{h}} + \boldsymbol{\phi})$$

Thus, considering a reference trajectory $h_{xr}(t), h_{yr}(t)$, the so-called open loop control \mathbf{T}_r is given by:

$$\mathbf{T}_r = \mathbf{C}_r + \mathbf{G}_r + M H_r^{-1}(\ddot{\mathbf{h}}_r + \boldsymbol{\phi}_r) \quad (3.24)$$

with

$$\mathbf{C}_r = \begin{pmatrix} -m_2 l_1 \dot{\theta}_{2r}^2 r_2 \sin \theta_{2r} - 2m_2 l_1 \dot{\theta}_{1r} \dot{\theta}_{2r} r_2 \sin \theta_{2r} \\ m_2 l_1 \dot{\theta}_{1r}^2 r_2 \sin \theta_{2r} \end{pmatrix}$$

$$\mathbf{G}_r = \begin{pmatrix} (m_2 l_1 + m_1 r_1) g \sin \theta_{1r} + m_2 r_2 g \sin(\theta_{1r} + \theta_{2r}) \\ m_2 r_2 g \sin(\theta_{1r} + \theta_{2r}) \end{pmatrix}$$

$$H_r = \begin{pmatrix} -h_{yr} & -l_2 \sin(\theta_{1r} + \theta_{2r}) \\ h_{xr} & l_2 \cos(\theta_{1r} + \theta_{2r}) \end{pmatrix}$$

$$\boldsymbol{\phi}_r = \begin{pmatrix} l_1 \dot{\theta}_{1r}^2 \cos \theta_{1r} + l_2 (\dot{\theta}_{1r} + \dot{\theta}_{2r})^2 \cos(\theta_{1r} + \theta_{2r}) \\ l_1 \dot{\theta}_{1r}^2 \sin \theta_{1r} + l_2 (\dot{\theta}_{1r} + \dot{\theta}_{2r})^2 \sin(\theta_{1r} + \theta_{2r}) \end{pmatrix}$$

$$\theta_{1r} = \arctan \left(\frac{h_{yr}}{h_{xr}} \right) - \arctan \left(\frac{l_2 \sin \theta_{2r}}{l_1 + l_2 \cos \theta_{2r}} \right)$$

$$\theta_{2r} = \arctan \left(\pm \frac{\sqrt{1 - \bar{h}_r^2}}{\bar{h}_r} \right)$$

$$\bar{h}_r = \frac{h_{xr}^2 + h_{yr}^2 - l_1^2 - l_2^2}{2l_1 l_2}$$

Then, the feedback control law ensuring tracking of the reference trajectory $h_{xr}(t), h_{yr}(t)$ is given by:

$$T = C + G + MH^{-1}(\boldsymbol{\phi} + \ddot{\mathbf{h}}_r - \Lambda_0^h \mathbf{e}_h - \Lambda_1^h \dot{\mathbf{e}}_h) \quad (3.25)$$

with

$$\Lambda_0^h = \begin{pmatrix} \lambda_{00}^h & 0 \\ 0 & \lambda_{01}^h \end{pmatrix}, \quad \Lambda_1^h = \begin{pmatrix} \lambda_{10}^h & 0 \\ 0 & \lambda_{11}^h \end{pmatrix}$$

where the λ_{ijh} are suitably chosen reals such that the closed loop error equation in \mathbf{e}_h is exponentially stable (it is thus sufficient to choose these as strictly positive reals).

Note that the difference between the previous tracking control law and the feedforward one given in (3.24) is of the form:

$$\begin{aligned} T - T_r &= C - C_r + G - G_r + \\ &\quad MH^{-1}(\boldsymbol{\phi} + \ddot{\mathbf{h}}_r) - MH_r^{-1}(\boldsymbol{\phi}_r + \ddot{\mathbf{h}}_r) - \\ &\quad MH^{-1}(\boldsymbol{\lambda}_h^T \mathbf{e}_h + \boldsymbol{\mu}_h^T \dot{\mathbf{e}}_h) \end{aligned} \quad (3.26)$$

which tends to zero when \mathbf{e}_h itself tends to zero.

In (3.25), one needs to compute H^{-1} (the matrix H being defined in (3.22)), which requires the determinant Δ_H

$$\Delta_H = l_2(h_x \sin(\theta_1 + \theta_2) - h_y \cos(\theta_1 + \theta_2))$$

to be non zero. From (3.18), we get

$$\Delta_H = l_1(h_x \sin \theta_1 - h_y \cos \theta_1)$$

Thus, when $\Delta_H = 0$, we get

$$\tan(\theta_1 + \theta_2) = \tan \theta_1$$

Or, what is the same

$$\theta_2 = 0, \quad \text{or} \quad \theta_2 = \pi$$

The first case yields the following end effector coordinates:

$$\begin{aligned} h_x &= (l_1 + l_2) \cos \theta_1 \\ h_y &= (l_1 + l_2) \sin \theta_1 \end{aligned}$$

Thus, the end effector with coordinates h_x, h_y remains on a circle centered at the origin and with radius $l_1 + l_2$, which corresponds to the arm being fully extended. The second case ($\theta_2 = \pi$) yields the end effector coordinates:

$$h_x = (l_1 - l_2) \cos \theta_1$$

$$h_y = (l_1 - l_2) \sin \theta_1$$

and the end effector with coordinates h_x, h_y remains on a circle centered at the origin and with radius $l_1 - l_2$, which corresponds to the arm fully folded.

When designing a reference trajectory, we shall avoid these two cases. Let us consider

$$h_{yr}(t) = \frac{h_{yf} - h_{yi}}{2} \left[1 + \tanh \left(\gamma (h_{xr}(t) - h_{x0}) \right) \right] \quad (3.27a)$$

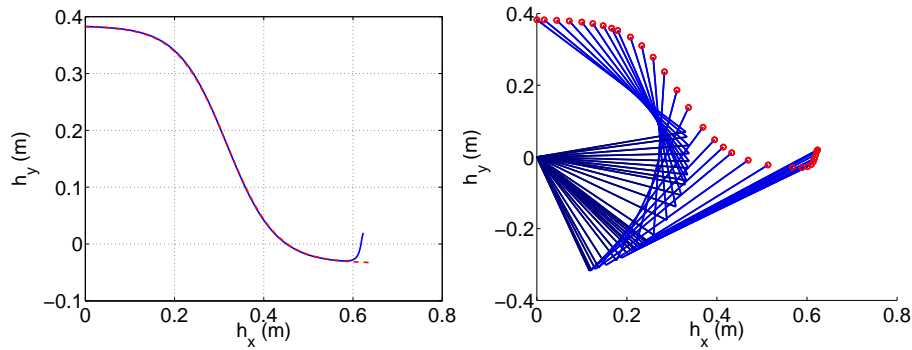
$$h_{xr}(t) = \frac{(h_{xf} - h_{xi})t}{T} + h_{xi} \quad (3.27b)$$

for $t \in [0, T]$, and for example:

$$h_{xi} = 0.8(l_1 + l_2), \quad h_{xf} = 0 \quad (3.28a)$$

$$h_{yi} = l_1 + 0.1l_2, \quad h_{yf} = -0.1l_1 \quad (3.28b)$$

The trajectory tracking is illustrated in Figure 3.2a, and the associated animation in Figure 3.2b. The corresponding control laws are



(a) End effector tracking: h_y versus h_x . (b) Animated tracking of the two link arm.

Figure 3.2: Trajectory tracking of a two link robot arm. In red and dashed the reference trajectory; in blue and solid, the actual (simulated) trajectory.

shown in Figures 3.3a and 3.3b. The tracking errors are plotted in

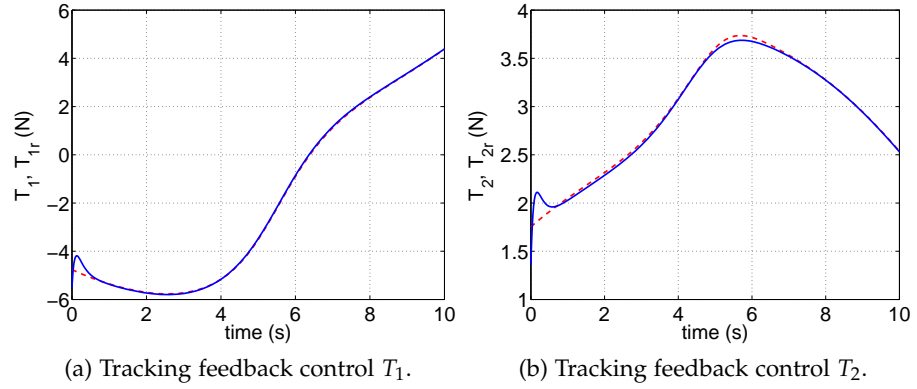


Figure 3.3: Two link arm tracking feedback control laws. In red and dashed the open loop (feedforward) law; in blue and solid, the feedback law.

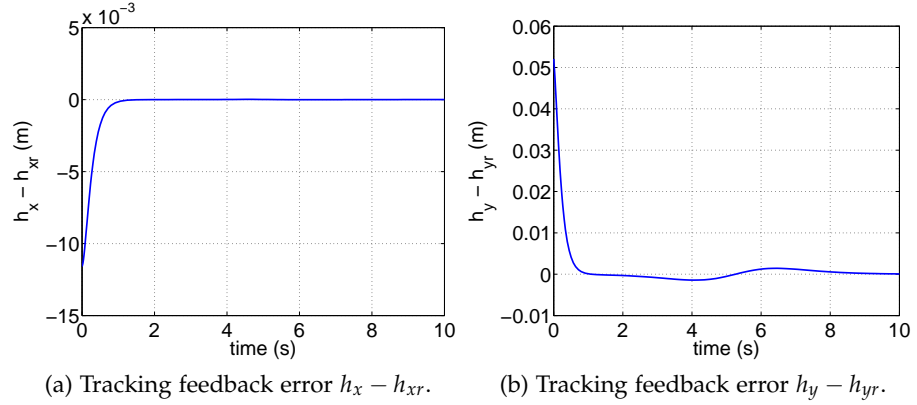


Figure 3.4: Two link arm tracking feedback errors.

Figures 3.4a and 3.4b.

The previous results can be applied to more complete and less simplistic models, as in e.g. (Friston, 2010); see also (Richardson et al., 2013).

3.4 EXTENSIONS OF DIFFERENTIAL FLATNESS

Several extensions of differentially flat systems can be envisioned. The recent paper (Aschenbrenner et al., 2013) reviews some of the most interesting ones. A Liouvillian closed structure \mathcal{D} will contain all the solutions of first order linear differential equations, an

extended Liouvillian one will contain all the solutions of linear differential equations and an existentially closed one all the solutions of algebraic differential equations. We shall give some rather elementary definitions below (see Appendix C.2, p. 62).

Definition 4 *The system*

$$\dot{x} = f(x, u)$$

with $x(t) \in \mathbb{R}^n$ and $u(t) \in \mathbb{R}^m$ is called Liouvillian (resp. extended Liouvillian, existentially closed) if there exists a set of variables, called a Liouvillian (resp. extended Liouvillian, existentially closed) output $y = (y_1, \dots, y_m)$ solution of

$$H(y, \dot{y}, \dots, y^{(r_y)}, x, u, \dot{u}, \dots, u^{(r_u)}) = 0, \quad r_y, r_u \in \mathbb{N} \quad (3.29)$$

with H linear of first order (resp. linear, polynomial) in its variables, such that

$$x = A(y, \dot{y}, \dots, y^{(\rho_x)})$$

$$u = B(y, \dot{y}, \dots, y^{(\rho_u)})$$

with q an integer, and such that the system equations

$$\begin{aligned} \frac{dA}{dt}(y, \dot{y}, \dots, y^{(q+1)}) = \\ f(A(y, \dot{y}, \dots, y^{(q)}), B(y, \dot{y}, \dots, y^{(q+1)})) \end{aligned}$$

are identically satisfied.

3.5 DIFFERENTIAL FLATNESS OF SOME OTHER NEURAL MASS MODELS

3.5.1 Jansen and Rit model

Brief recall of the model

Consider the Jansen and Rit model, as depicted in (Pinotsis et al., 2012):

$$\ddot{v}_1 + 2\kappa_e \dot{v}_1 + \kappa_e^2 v_1 = \kappa_e m_e (w_{13} F(v_3) + u) \quad (3.30a)$$

$$\ddot{v}_2 + 2\kappa_i \dot{v}_2 + \kappa_i^2 v_2 = \kappa_i m_i w_{23} F(v_3) \quad (3.30b)$$

$$\ddot{v}_3 + 2\kappa_e \dot{v}_3 + \kappa_e^2 v_3 = \kappa_e m_e (w_{31} F(v_1) + w_{32} F(v_2)) \quad (3.30c)$$

$$y = v_3 \quad (3.30d)$$

These equations respectively depict the following populations: excitatory stellate, inhibitory and excitatory. The signification of the various variables and parameters are the following:

- v_i Expected depolarization in the i -th population
- $w_{ij}F(v_j)$ presynaptic input to the i -th population from the j^{th} one
- $F(v_j)$ Sigmoid function of the postsynaptic depolarization
- w_{ij} Intrinsic connection strength between the populations j and i
- m_i, m_e Maximum postsynaptic responses
- κ_e, κ_i Rate constants of postsynaptic filtering
- u Exogenous input
- y Endogenous output

The choice made in (Pinotsis et al., 2012) for the sigmoid function F is the logistic function

$$F(v) = \frac{1}{1 + e^{-\beta(v-v_T)}}$$

whose derivative and inverse are:

$$F' = \beta F(F - 1), \quad \text{and} \quad F^{-1}(\eta) = \phi(\eta) = v_T + \frac{1}{\beta} \ln \frac{\eta}{\eta - 1}$$

Let d_i, d_e be the differential operators

$$d_i = \frac{d^2}{dt^2} + 2\kappa_i \frac{d}{dt} + \kappa_i^2 = \left(\frac{d}{dt} + \kappa_i \right)^2$$

$$d_e = \left(\frac{d}{dt} + \kappa_e \right)^2$$

Then, the previous model (3.30) can be written as

$$d_e v_1 = \kappa_e m_e (w_{13} F(v_3) + u) \quad (3.31a)$$

$$d_i v_2 = \kappa_i m_i w_{23} F(v_3) \quad (3.31b)$$

$$d_e v_3 = \kappa_e m_e (w_{31} F(v_1) + w_{32} F(v_2)) \quad (3.31c)$$

$$y = v_3 \quad (3.31d)$$

Differential flatness of the model

A flat output of the model (3.30) is v_2 . Indeed, after equation (3.30b), one gets v_3 :

$$v_3 = \phi \left(\frac{1}{\kappa_i m_i w_{23}} (\ddot{v}_2 + 2\kappa_i \dot{v}_2 + \kappa_i^2 v_2) \right) \quad (3.32)$$

Then, after (3.30c)

$$w_{31}F(v_1) = w_{32}F(v_2) + \frac{1}{\kappa_e m_e} (\ddot{v}_3 + 2\kappa_e \dot{v}_3 + \kappa_e^2 v_3)$$

Hence the expression for v_1 :

$$v_1 = \phi \left[\frac{w_{32}}{w_{31}} F(v_2) + \frac{1}{\kappa_e m_e w_{31}} (\ddot{v}_3 + 2\kappa_e \dot{v}_3 + \kappa_e^2 v_3) \right] \quad (3.33)$$

And, using (3.30a), the expression for u :

$$u = -w_{13}F(v_3) + \frac{1}{\kappa_e m_e} (\ddot{v}_1 + 2\kappa_e \dot{v}_1 + \kappa_e^2 v_1) \quad (3.34)$$

The Figure 3.5 below outlines the compartmental like model underlying the model (3.30). The bold arrows in this Figure enables one,

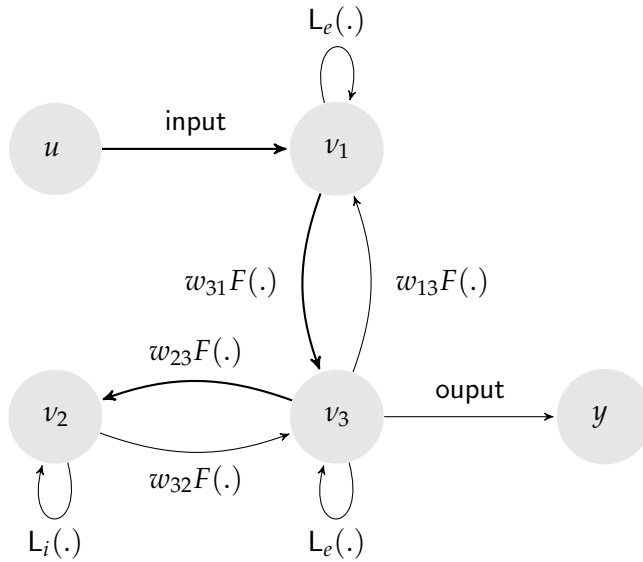


Figure 3.5: The Jansen and Rit Model.

by reversing the arrows, to reveal the differential flatness character of the model: v_3 is obtained from v_2 by reversing the arrow $(v_3) \rightarrow (v_2)$ (yielding equation (3.32)) ; then, v_1 is obtained from v_3 (and v_2) by reversing the arrow $(v_1) \rightarrow (v_3)$ (yielding equation (3.33)) ; finally u is obtained from v_1 (and v_3) by reversing the arrow $(u) \rightarrow (v_1)$ (yielding equation (3.34)).

Extended Liouvillian character of the model

Since the output of interest considered in (Pinotsis et al., 2012) is v_3 , we can investigate how the model can be parametrized by this variable. The variable v_2 can be obtained from v_3 by integrating the differential equation (3.30b) in v_2 (which is linear in this variable). Indeed, (3.30b) can be rewritten as:

$$\frac{d}{dt} \begin{pmatrix} v_2 \\ \dot{v}_2 \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ -\kappa_i^2 & -2\kappa_i \end{pmatrix} \begin{pmatrix} v_2 \\ \dot{v}_2 \end{pmatrix} + \begin{pmatrix} 0 \\ \kappa_i m_i w_{23} F(v_3) \end{pmatrix}$$

or, in matrix form

$$\dot{V} = AV + U, \quad \text{with} \\ V = \begin{pmatrix} v_2 \\ \dot{v}_2 \end{pmatrix}, \quad A = \begin{pmatrix} 0 & 1 \\ -\kappa_i^2 & -2\kappa_i \end{pmatrix}, \quad U = \begin{pmatrix} 0 \\ \kappa_i m_i w_{23} F(v_3) \end{pmatrix}$$

The general solution of this last equation is well known to be

$$V = V(0)e^{At} + \int_0^t e^{A(t-\tau)} U(\tau) d\tau$$

One has

$$e^{At} = \begin{pmatrix} (1 + \kappa_i t)e^{-\kappa_i t} & te^{-\kappa_i t} \\ -\kappa_i^2 e^{-\kappa_i t} & (1 - \kappa_i t)e^{-\kappa_i t} \end{pmatrix}$$

Thus, v_2 is given by

$$v_2 = v_{20}((1 + \kappa_i t)e^{-\kappa_i t}) + \dot{v}_{20}te^{-\kappa_i t} + \kappa_i m_i w_{23} \int_0^t (t - \tau)e^{-\kappa_i(t-\tau)} F(v_3(\tau)) d\tau \quad (3.35)$$

where $v_{20} = v_2(0)$, $\dot{v}_{20} = \dot{v}_2(0)$. Then, the two other variables are obtained as in (3.33)–(3.34):

$$v_1 = \phi \left[\frac{w_{32}}{w_{31}} F(v_2) + \frac{1}{\kappa_e m_e w_{31}} (\dot{v}_3 + 2\kappa_e \dot{v}_3 + \kappa_e^2 v_3) \right] \quad (3.36a)$$

$$u = -w_{13} F(v_3) + \frac{1}{\kappa_e m_e} (\ddot{v}_1 + 2\kappa_e \dot{v}_1 + \kappa_e^2 v_1) \quad (3.36b)$$

Recalling d_i, d_e the differential operators

$$d_i = \left(\frac{d}{dt} + \kappa_i \right)^2 \quad d_e = \left(\frac{d}{dt} + \kappa_e \right)^2$$

The previous equations (3.35)–(3.36) can be rewritten as:

$$\nu_2 = \mathbf{d}_i^{-1}(\kappa_i m_i w_{23} F(\nu_3)) \quad (3.37a)$$

$$\nu_1 = \phi \left[\frac{w_{32}}{w_{31}} F(\nu_2) + \frac{1}{\kappa_e m_e w_{31}} \mathbf{d}_e \nu_3 \right] \quad (3.37b)$$

$$u = -w_{13} F(\nu_3) + \frac{1}{\kappa_e m_e} \mathbf{d}_e \nu_1 \quad (3.37c)$$

Thus, the model is extended Liouvillian and an extended Liouvillian output is ν_3 .

Part II

NEURAL FIELD POPULATION MODELS

3.5.2 General case model

We consider spatially distributed network models, such as the ones considered in Chapter 8 of (Ermentrout and Terman, 2010), Subsection 8.4, p. 223, and Chapter 12, Subsection 12.3.1, p. 376 or in Chapter 6 of (Bressloff, 2014), Subsection 6.5, p. 264 (as well as Subsection 2.5, p. 14 of (Bressloff, 2012)).

We can consider the so-called activity-based neural field model:

$$\tau_{sy} \frac{\partial v(t, x)}{\partial t} = -v(t, x) + F(I_r w(x) *_{\mathbf{x}} v(t, x)) h(v(t, x)) + u(t, x)$$

or, in a slightly more compact way

$$\tau_{sy} \frac{\partial v}{\partial t} = -v + F(I_r w(x) *_{\mathbf{x}} v) h(v) + u \quad (3.38)$$

And a slightly more general case

$$\tau_{sy} \frac{\partial v}{\partial t} = -v + F \left(I_r \int_{\Omega_x} w(x - \xi) v(t, \xi) d\xi \right) h(v) + u \quad (3.39)$$

Alternately, we can consider the slightly different model

$$\tau_{sy} \frac{\partial v}{\partial t} = -v + I_r w(x) *_{\mathbf{x}} F(v(t, x)) h(v) + u \quad (3.40)$$

And its slight generalization

$$\tau_{sy} \frac{\partial v}{\partial t} = -v + I_r \int_{\Omega_x} w(x - \xi) F(v(t, \xi)) d\xi h(v) + u \quad (3.41)$$

3.5.3 Parameters and variables assumptions

The various parameters and functions staisfy the following:

- The spatial variable is three dimensional, i.e. $\Omega_x \subset \mathbb{R}^3$.
- The parameter τ_{sy} is a constant.
- If the synapses are saturating, then $h(s) = 1 - s$, otherwise, $h(s) = 1$.
- The function F is a sigmoid type function (see A.3, p. 54).
- The control $u(t, r)$ has a spatial compact support Ω_u .
- The neuron interaction strength function $w(r)$ is symmetric, nonnegative, integrates to 1 over the whole line and is rapidly decaying at infinity:

$$\exists M \in \mathbb{R}^3, \exists \alpha \in \mathbb{R}^+, \forall x \in \mathbb{R}^3 \text{ s.t. } \|x\| > M, \|w(x)\| < \|x\|^{-\alpha}$$

Acronym	Name	Function w
(Wdorg)	<i>Dirac at the origin</i>	δ_0
(Wdnor)	<i>Dirac not at the origin</i>	δ_{x_0}
(Wsofd)	<i>Sum of Diracs</i>	$\sum_{i=1}^N a_i \delta_{x_i}$
(Wsexp)	<i>Single exponential</i>	$e^{-ax} H(t)$
(Wmexp)	<i>Multiple exponential</i>	$\sum_{i=1}^N e^{-a_i x} H(t)$
(Wgaus)	<i>Gaussian</i>	e^{-x^2/σ^2}
(Waexp)	<i>Absolute exponential</i>	$e^{- x /2}$
(Wdosc)	<i>Decaying oscillatory</i>	$e^{-b x }(b \sin x + \cos x)$
(Wfhat)	<i>Flat hat shaped</i>	$\text{rect}(x/\chi)$
(Wmhat)	<i>Mexican hat</i>	$\Gamma_1 e^{-\gamma_1 x} - \Gamma_2 e^{-\gamma_2 x}$ or $e^{-c_1 x^2/\sigma_1^2}/\sigma_1^2 - e^{-c_2 x^2/\sigma_2^2}/\sigma_2^2$
(Wwhat)	<i>Wizard hat</i>	$(1/4)(1 - x)e^{- x }$
(Wcomp)	<i>Compact support</i>	w has compact support

Table 3.1: Neuron interaction strength functions examples.

3.5.4 Neuron interaction strength examples

Some typical examples of such w functions are shown in Table 3.1.

The graphical representations of some of the above mentioned neuron interaction strength functions are given in Figures 3.6 and 3.7.

3.5.5 Simplification hypotheses

We shall make the following simplifying assumptions:

(H1) We consider a spherical symmetric case, which boils down to the unidimensional case where $\Omega_x = [a, b]$ in r and $r = \|x\|$.

(H2) We consider F equal to a heaviside:

$$\forall \xi \in \mathbb{R}^-, \quad F(\xi) = 0, \quad \forall \xi \in \mathbb{R}^+, \quad F(\xi) = 1$$

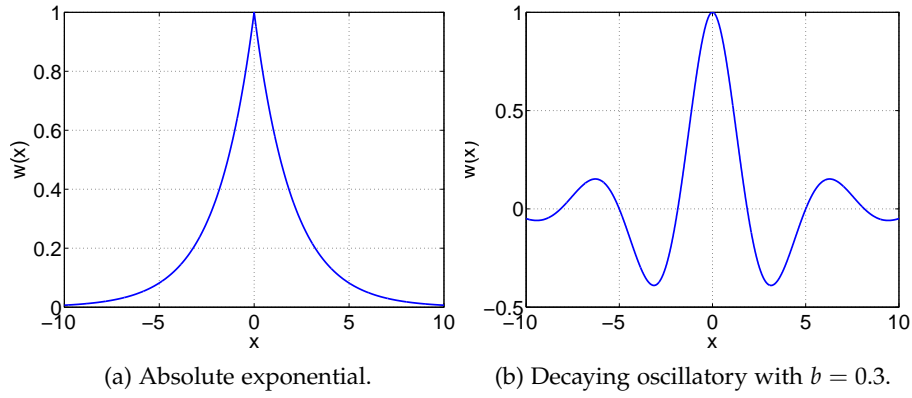


Figure 3.6: Examples of neuron interaction strength functions.

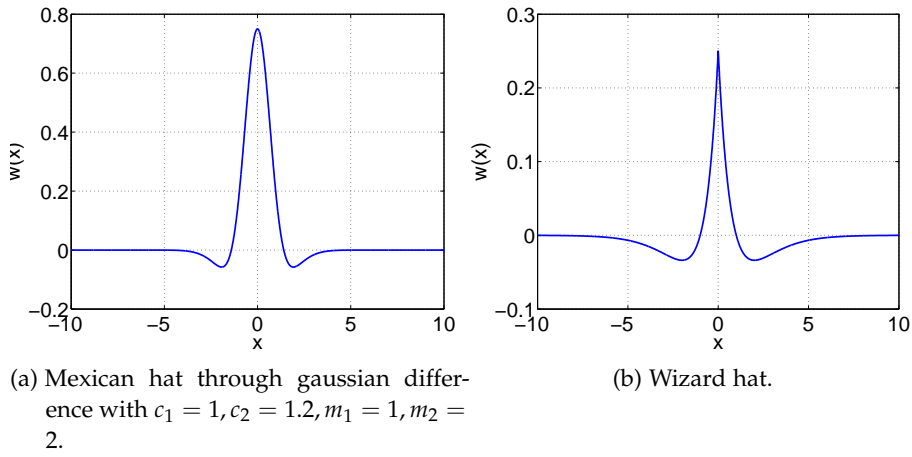


Figure 3.7: Examples of neuron interaction strength functions.

(H3) We consider that the synapses are not saturating, hence $h(s) = 1$.

3.5.6 Pointwise neuron interaction strength models

Dirac at the origin case

We consider that w is a single spatial Dirac:

$$w(r) = \delta_0$$

We then obtain the following system

$$\dot{v}(t, r) = -av(t, r) + \delta_0 *_r (\mathbb{1}_{[a, b]} v(t, r)) + u(t, r)$$

Or, in other words

$$\dot{v}(t, r) = -av(t, r) + \mathbb{1}_{[a, b]}(r) v(t, r) + u(t, r) \quad (3.42)$$

Dirac not at the origin case

We consider that w is a single spatial Dirac:

$$w(t, r) = \delta_{r_0}$$

with $r_0 \in [a, b]$. We then obtain the following system

$$\dot{v}(t, r) = -av(t, r) + \delta_{r_0} *_r (\mathbb{1}_{[a, b]} v(t, r)) + u(t, r) \quad (3.43)$$

Or, in other words

$$\dot{v}(t, r) = -av(t, r) + \mathbb{1}_{[a, b]}(r - r_0) v(t, r - r_0) + u(t, r) \quad (3.44)$$

3.5.7 Exponential type neuron interaction strength

A single exponential

Consider that the neuron interaction strength w satisfies a linear differential equation:

$$w'(r) = -aw(r)$$

The model is the following

$$\tau_{sy} \frac{\partial v(t, r)}{\partial t} = -v(t, r) + I_r w(r) *_r v(t, r) + u(t, r) \quad (3.45)$$

Hence, the convolution part is

$$w(r) *_r v(t, r) = \frac{1}{I_r} \left(\tau_{sy} \frac{\partial v(t, r)}{\partial t} + v(t, r) - u(t, r) \right)$$

And is spatial derivative is

$$\begin{aligned} w'(r) *_r v(t, r) &= \frac{1}{I_r} \left(\tau_{sy} \frac{\partial^2 v(t, r)}{\partial t \partial r} + \frac{\partial v(t, r)}{\partial r} + \frac{\partial u(t, r)}{\partial r} \right) \\ &= -aw(r) *_r v(t, r) \\ &= \frac{-a}{I_r} \left(\tau_{sy} \frac{\partial v(t, r)}{\partial t} + v(t, r) - u(t, r) \right) \end{aligned}$$

Hence, v satisfies the following partial differential equation:

$$\tau_{sy} \partial_t \partial_r v(t, r) = -(a\tau_{sy} \partial_t + \partial_r + a)v(t, r) - (\partial_r + a)u(t, r)$$

A more general case

In (Coombes et al., 2014), Chapter 5, “PDE Methods for Two-Dimensional Neural Fields” by Carlo R. Laing, the case of neuron interaction strength w with a rational Fourier transform is considered:

$$\mathcal{F}(w)(\xi) = \tilde{w}(\xi) = \int_{-\infty}^{+\infty} w(\tau) e^{-j\xi\tau} d\tau = \frac{p(\xi^2)}{q(\xi^2)}$$

with p and q two polynomials. Then, equation (3.40) with $h(s) = 1$:

$$\tau_{sy} \frac{\partial v(t, x)}{\partial t} = -v(t, x) + I_r w(x) *_x F(v(t, x)) + u(t, x)$$

becomes

$$\begin{aligned} (\tau_{sy} \partial_t + 1) \tilde{v}(t, \xi) &= I_r \tilde{w}(\xi) \tilde{F}(v)(t, \xi) + \tilde{u}(t, \xi) \\ &= I_r \frac{p(\xi^2)}{q(\xi^2)} \tilde{F}(v)(t, \xi) + \tilde{u}(t, \xi) \end{aligned}$$

Thus, through multiplication by $q(\xi^2)$:

$$(\tau_{sy} \partial_t + 1) q(\xi^2) \tilde{v}(t, \xi) = I_r p(\xi^2) \tilde{F}(v)(t, \xi) + q(\xi^2) \tilde{u}(t, \xi)$$

which yields, in the spatial domain:

$$(\tau_{sy} \partial_t + 1) q(\partial_x^2) v(t, x) = I_r p(\partial_x^2) F(v)(t, x) + q(\partial_x^2) u(t, x)$$

The Fourier transforms of some of the neuron interaction strength functions of Table 3.1, p. 40 are shown in Table 3.2.

3.6 A JANSEN AND RIT NEURAL FIELD MODEL

Let us consider, after (Pinotsis et al., 2012), the following neural field Jansen and Rit model

$$\dot{v}_1 + 2\kappa_e \dot{v}_1 + \kappa_e^2 v_1 = \kappa_e m_e (\mu_1 + u) \quad (3.46a)$$

$$\dot{v}_2 + 2\kappa_i \dot{v}_2 + \kappa_i^2 v_2 = \kappa_i m_i \mu_2 \quad (3.46b)$$

$$\dot{v}_3 + 2\kappa_e \dot{v}_3 + \kappa_e^2 v_3 = \kappa_e m_e \mu_3 \quad (3.46c)$$

$$\partial_t^2 \mu_1 - \sigma^2 \partial_x^2 \mu_1 + 2\sigma \beta_{13} \partial_t \mu_1 + \sigma^2 \beta_{13}^2 \mu_1 = \phi_{13}(v_3) \quad (3.46d)$$

$$\partial_t^2 \mu_2 - \sigma^2 \partial_x^2 \mu_2 + 2\sigma \beta_{23} \partial_t \mu_2 + \sigma^2 \beta_{23}^2 \mu_2 = \phi_{23}(v_3) \quad (3.46e)$$

$$\partial_t^2 \mu_3 - \sigma^2 \partial_x^2 \mu_3 + 2\sigma \beta_{31} \partial_t \mu_3 + \sigma^2 \beta_{31}^2 \mu_3 = \psi_{31}(v_1, v_2) \quad (3.46f)$$

Name	Function/Distribution	Fourier transform
<i>Dirac at the origin</i>	δ_0	1
<i>Dirac not at the origin</i>	δ_{x_0}	$e^{-j\tilde{\zeta}x_0}$
<i>Sum of Diracs</i>	$\sum_{i=1}^N a_i \delta_{x_i}$	$\sum_{i=1}^N a_i e^{-j\tilde{\zeta}x_i}$
<i>Flat hat shaped</i>	$\text{rect}\left(\frac{x}{\chi}\right)$	sinc
<i>Gaussian</i>	$e^{-\frac{x^2}{\sigma^2}}$	$\frac{\sigma}{\sqrt{2}} e^{-\frac{\sigma^2 \tilde{\zeta}^2}{4}}$
<i>Absolute exponential</i>	$e^{-a x }$	$\frac{2a}{a^2 + \tilde{\zeta}^2}$
<i>Decaying oscillatory</i>	$e^{-b x } (b \sin x + \cos x)$	$\frac{4b(b^2 + 1)}{\tilde{\zeta}^4 + 2(b^2 - 1)\tilde{\zeta}^2 + (b^2 + 1)^2}$
<i>Mexican hat</i>	$\frac{\Gamma_1 e^{-\gamma_1 x } - \Gamma_2 e^{-\gamma_2 x }}{\Gamma_2 e^{-\gamma_2 x }}$	$2 \frac{\Gamma_1 \gamma_1 (\gamma_2 + \tilde{\zeta}^2) - \Gamma_2 \gamma_2 (\gamma_1^2 + \tilde{\zeta}^2)}{(\gamma_1^2 + \tilde{\zeta}^2)(\gamma_2^2 + \tilde{\zeta}^2)}$
<i>Wizard hat</i>	$\frac{(1 - x)e^{- x }}{4}$	$\frac{\tilde{\zeta}^2}{(1 + \tilde{\zeta}^2)^2}$

Table 3.2: Fourier tranforms of some neuron interaction strength functions.

with

$$\begin{aligned}\phi_{ij}(\nu_j) &= \alpha_{ij} (\sigma^2 F(\nu_j) + \sigma F'(\nu_j)) \\ \psi_{31}(\nu_1, \nu_2) &= \alpha_{31} (\sigma^2 \beta_{31} (F(\nu_1) - F(\nu_2)) + \sigma (F'(\nu_1) - F'(\nu_2)))\end{aligned}$$

Let us recall the differential operators d_i and d_e and introduce D_1 , D_2 , D_3 :

$$d_i = \left(\frac{d}{dt} + \kappa_i \right)^2 \quad d_e = \left(\frac{d}{dt} + \kappa_e \right)^2$$

$$\begin{aligned}D_1 &= \partial_t^2 - \sigma^2 \partial_x^2 + 2\sigma \beta_{13} \partial_t + \sigma^2 \beta_{13}^2 \\ D_2 &= \partial_t^2 - \sigma^2 \partial_x^2 + 2\sigma \beta_{23} \partial_t + \sigma^2 \beta_{23}^2, \\ D_3 &= \partial_t^2 - \sigma^2 \partial_x^2 + 2\sigma \beta_{31} \partial_t + \sigma^2 \beta_{31}^2\end{aligned}$$

The model (3.46) can then be rewritten as

$$d_e \nu_1 = \kappa_e m_e (\mu_1 + u) \quad (3.47a)$$

$$d_i \nu_2 = \kappa_i m_i \mu_2 \quad (3.47b)$$

$$d_e \nu_3 = \kappa_e m_e \mu_3 \quad (3.47c)$$

$$D_1 \mu_1 = \phi_{13}(\nu_3) \quad (3.47d)$$

$$D_2 \mu_2 = \phi_{23}(\nu_3) \quad (3.47e)$$

$$D_3 \mu_3 = \psi_{31}(\nu_1, \nu_2) \quad (3.47f)$$

The Figure 3.8 below outlines the compartmental like model underlying the model (3.47).

The variable ν_2 is a Liouvillian output. Indeed, μ_2 is obtained through ν_2 using equation (3.47b):

$$\mu_2 = \frac{1}{\kappa_i m_i} d_i \nu_2 \quad (3.48)$$

Then ν_3 is obtained via μ_2 with the help of (3.47e)

$$\nu_3 = \psi(D_2 \mu_2) \quad (3.49)$$

where ψ_2 denotes the inverse function of $\alpha_{23}(\sigma^2 \beta_{23} F + \sigma F')$. After, μ_3 is derived from ν_3 through (3.47c)

$$\mu_3 = \frac{1}{\kappa_e m_e} d_e \nu_3 \quad (3.50)$$

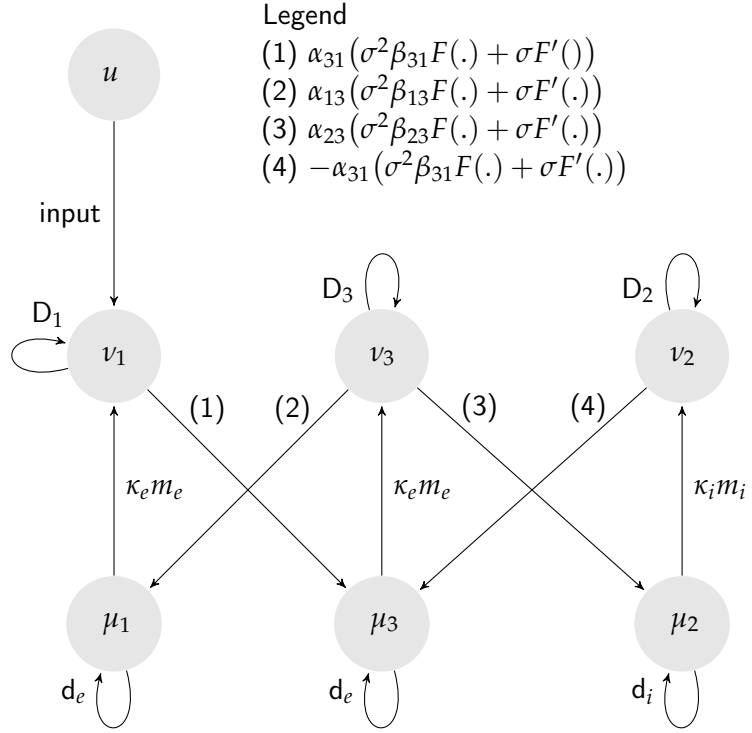


Figure 3.8: A Jansen and Rit neural field model.

The variable μ_3 and v_2 yields v_1 through (3.47f)

$$v_1 = D_3 \mu_3 + \alpha_{31}(\sigma^2 \beta_{31} F(v_2) + \sigma F'(v_2)) \quad (3.51)$$

where ψ_3 is the inverse function of $\alpha_{31}(\sigma^2 \beta_{31} F + \sigma F')$. By integrating the wave equation in the equation (3.47d) we get μ_1 from v_3

$$\mu_1 = \alpha_{13} D_1^{-1}(\sigma^2 \beta_{13} F(v_3) + \sigma F'(v_3)) \quad (3.52)$$

At last, the control input u can be derived through (3.47a) from μ_1 and v_1

$$u = \frac{1}{\kappa_e m_e} d_e v_1 - \mu_1 \quad (3.53)$$

BIBLIOGRAPHY

- M. Aschenbrenner, L. van den Dries, and J. van der Hoeven. Towards a model theory for transseries. *Notre Dame Journal of Formal Logic*, 54:279–310, 2013.
- M. Aschenbrenner and L. van den Dries. Liouville closed h-fields. *Journal of Pure and Applied Algebra*, 197:83–139, 2005a.
- M. Aschenbrenner and L. van den Dries. *Analyzable Functions and Applications*, volume 373 of *Contemp. Math.*, chapter Asymptotic differential algebra, pages 49–85. Amer. Math. Soc., Providence, RI, 2005b.
- P.C. Bressloff. Spatiotemporal dynamics of continuum neural fields. *J. Phys. A: Math. Theor.*, 45, 2012.
- P.C. Bressloff. *Waves in Neural Media – From Single Neurons to Neural Fields*. Springer, New York, 2014.
- R. Brette and W. Gerstner. Adaptive exponential integrate-and-fire model as an effective description of neuronal activity. *J. Neurophysiol.*, 94:3637–3642, 2005.
- S. Coombes, P. beim Graben, R. Potthast, and J. Wright. *Neural Fields – Theory and Applications*. Springer, Berlin, 2014.
- P. Dayan and L. F. Abbott. *Theoretical Neuroscience: Computational and Mathematical Modeling of Neural Systems*. The MIT Press, Cambridge, Massachussets, 2005.
- B. Dumitrescu. *Positive Trigonometric Polynomials and Signal Processing Applications*. Springer, Dordrecht, The Netherlands, 2007.
- G.B. Ermentrout and D.H. Terman. *Mathematical Foundations of Neuroscience*. Springer, New York, 2010.
- M. Fliess, J. Lévine, P. Martin, and P. Rouchon. Flatness and defect of non-linear systems: introductory theory and applications. *Internat. J. Control*, 61:1327–1361, 1995.

- M. Fliess, H. Mounier, P. Rouchon, and J. Rudolph. Tracking control of a vibrating string with an interior mass viewed as delay system. *ESAIM Control Optim. Calc. Var.*, 3:315–321, 1998.
- K. Friston. The free-energy principle: a unified brain theory? *Nat. Rev. Neurosci.*, 11:127–138, 2010.
- K. Friston. A free energy principle for biological systems. *Entropy*, 14:2100–2121, 2012.
- H. Haken. *Brain Dynamics – An Introduction to Models and Simulation*. Springer-Verlag, Berlin, 2008.
- E.M. Izhikevich. Simple model of spiking neurons. *IEEE Transactions on Neural Networks*, 14:1569–1572, 2003.
- E.M. Izhikevich. Which model to use for cortical spiking neurons? *IEEE Transactions on Neural Networks*, 15:1063–1070, 2004.
- E.M. Izhikevich. Hybrid spiking models. *Phil. Trans. R. Soc. A*, 368:5061–5070, 2010.
- P. Martin and P. Rouchon. Systèmes plats : planification et suivi de trajectoires. In *Actes des Journées Nationales de Calcul Formel*, pages 197–276, 2008. URL <http://jncf2008.loria.fr/jncf2008.pdf>.
- K.D. Miller and F. Fumarola. Mathematical equivalence of two common forms of firing-rate models of neural networks. *Neural Comput.*, 24:25–31, 2012.
- K. Friston and D.A. Pinotsis. Xx.
- D.A. Pinotsis, R.J. Moran, and K.J. Friston. Dynamic causal modeling with neural fields. *Neuroimage*, 59:1261–1274, 2012.
- M.J. Richardson, M.A. Riley, and K. Shockley, editors. *Progress in Motor Control – Neural, Computational and Dynamic Approaches*, volume 782 of *Advances in experimental medicine and biology*. Springer, Dordrecht, 2013.
- P. Rouchon. Motion planning, equivalence, infinite dimensional systems. *Int. J. of Applied Mathematics and Computer Science*, 11, 2001.
- J. Touboul. Importance of the cutoff value in the quadratic adaptive integrate-and-fire model. *Neural Comput.*, 21:2114–2122, 2009.

- J. van der Hoeven. *Transseries and real differential algebra*, volume 1888 of *Lecture Notes in Mathematics*. Springer-Verlag, 2006.
- F. Woittennek and H. Mounier. Controllability of networks of spatially one-dimensional second order PDE – an algebraic approach. *Siam J. Contr.*, 48:3882–3902, 2010.

Part III

APPENDIX



NOTATIONS, TRANSFORMS AND SIGMOID FUNCTIONS

A.1 FUNCTIONS AND DISTRIBUTIONS

- The distribution $H(\eta)$ is the *Heaviside* distribution:

$$H(\eta) = \begin{cases} 0 & \text{if } \eta \leq 0 \\ 1 & \text{if } \eta > 0 \end{cases}$$

- The function $\text{sinc}(\eta)$ is the *cardinal sine*:

$$\text{sinc}(\eta) = \frac{\sin(\eta)}{\eta}$$

- The distribution $\text{rect}(\eta)$ is the *rectangular pulse* of width 1:

$$\text{rect}(\eta) = \begin{cases} 0 & \text{if } |\eta| \geq \frac{1}{2} \\ 1 & \text{if } |\eta| < \frac{1}{2} \end{cases}$$

- The *boxcar* distribution (rectangular pulse of width ρ and centered on η_0):

$$\text{rect}\left(\frac{t - t_0}{\rho}\right) = H(\eta - \eta_0 + \frac{\rho}{2}) - H(t - t_0 - \frac{\rho}{2})$$

- The *linear rectifier* function is

$$|\eta|_+ = \begin{cases} 0 & \text{when } v \leq 0 \\ v & \text{when } v > 0 \end{cases}$$

A.2 TRANSFORMS

Consider a function $f(t, x)$ from $\mathbb{R} \times \Omega_x$ to \mathbb{R} , where $\Omega_x \subseteq \mathbb{R}^3$.

- The function $\tilde{f}(t, \xi)$ will designate the *spatial Fourier transform* of f , i.e.

$$\tilde{f}(t, \xi) = \mathcal{F}(f)(t, \xi) = \int_{-\infty}^{+\infty} f(t, x) e^{-i\xi x} dx$$

- The function $\hat{f}(s, x)$ will designate the *temporal Laplace transform* of f , i.e.

$$\hat{f}(s, x) = \mathcal{L}(f)(s, x) = \int_{-\infty}^{+\infty} f(t, x) e^{-st} dt$$

A.3 SIGMOID FUNCTIONS

The following functions F will be in particular used for the firing rate. Thus $F(\xi)$ designates a spike rate and ξ a stimulus intensity. A sigmoid function $F : \mathbb{R} \rightarrow \mathbb{R}$ is such that

$$\begin{aligned} F(0) = F'(0) &= 0, & F(1) &= 1, & F'(1) &= 0 \\ \forall \xi \in \mathbb{R}, \xi < 0 & \quad F(\xi) = 0, \\ \forall \xi \in \mathbb{R}, \xi > 1 & \quad F(\xi) = 1 \end{aligned}$$

The following lists some of the most used sigmoid functions (see, e.g. (Ermentrout and Terman, 2010), Section 11.3, p. 345; (Bressloff, 2014), pp. 9, 22, 254, 373, (Haken, 2008), pp. 14, 252).

- The *Heaviside*.

$$F(x) = F_0 H(\xi - \xi_0)$$

- The *Piecewise linear* function.

$$F(x) = \begin{cases} 0 & \text{if } \xi < \xi_0 \\ \beta(\xi - \xi_0) & \text{if } \xi_0 \leq \xi < \xi_0 + 1/\beta \\ 0 & \text{if } \xi > \xi_0 + 1/\beta \end{cases}$$

- The *Logistic* function.

$$F(x) = \frac{1}{1 + e^{-\beta(x-x_T)}}$$

one has

$$F' = \beta F(F - 1)$$

$$F^{-1}(\eta) = \phi(\eta) = x_T + \frac{1}{\beta} \ln \frac{\eta}{\eta - 1}$$

- The *Traub Model*.

$$F(\xi) = \frac{1}{1 + e^{\frac{-\xi - \beta}{\alpha}}}$$

- The *Hyperbolic tangent* function.

$$F(\xi) = F_0(1 + \tanh(\alpha\xi))$$

- The *Square root* function.

$$F(\xi) = F_0 \sqrt{\xi - \xi_T}$$

- The *Noisy firing rate* function.

$$F(\xi) = \sqrt{\frac{\xi - \xi_T}{1 - e^{\frac{-(\xi - \xi_T)}{\beta}}}}$$

Here, β is a measure of the noise, and when β tends to zero, the function approaches a pure square root model.

- The *Mean firing rate with flexible shape* function (see (Coombes et al., 2014), p. 371).

$$F(\xi) = F_m - \frac{F_m}{(1 + e^{\sqrt{2} \frac{\xi - \mu}{\sigma}})^{\kappa}}$$

- The *Naka-Rushton* functions (alternately called Hill functions).

$$F(\xi) = \begin{cases} \frac{r\xi^n}{\xi^n + \theta^n} & \text{if } \xi \leq 0 \\ 0 & \text{if } \xi > 0 \end{cases}$$

where r is the maximum spike rate and θ is the value of the stimulus intensity for which F reaches half its maximum. The exponent n is a measure of the steepness of the $F(\xi)$ curve. Typical values matching experimental data range from 1.4 to 3.4. The function

$$F(\xi) = 1 - \frac{\xi^n}{\xi^n + \theta} = \frac{\theta^n}{\xi^n + \theta^n}$$

is also used.

- The *Algebraic sigmoid function*.

$$F(\xi) = \frac{\xi}{\sqrt{1 + \xi^2}}$$

This function has the inverse

$$F^1(\eta) = \frac{\eta}{\sqrt{1 - \eta^2}}$$

B

SOME FLATNESS SIMPLE CRITERIA

There does not exist, at the time of this writing, a general criterion for checking flatness, neither for building flat outputs in a constructive manner. Nevertheless, some peculiar cases are to be noticed.

B.1 NECESSARY AND SUFFICIENT CONDITIONS IN PECULIAR CASES

Proposition 2 *Any static state feedback linearizable system is flat.*

See below (Subsection B.3, p. 58) for a static state feedback linearizability criterion for affine input systems.

Proposition 3 (Charlet, Levine and Marino, 1989) *For systems with a single input, dynamic feedback linearization implies static feedback linearization.*

Proposition 4 (Charlet, Levine and Marino, 1989) *A dynamics affine in the input with n states and $n - 1$ inputs is flat as soon as it is controllable (strongly accessible).*

Recall that a dynamics is called affine in the input if it is of the form

$$\dot{x} = f_0(x) + \sum_{i=1}^{n-1} g_i(x)u_i$$

A dynamics with $x \in \mathcal{X} \subseteq \mathbb{R}^n$ is strongly accessible if, for all $x \in \mathcal{X}$, there exists a $T > 0$ such that

$$\text{intR}(x) \neq \emptyset$$

where intS denotes the interior of the set S and $R(x)$ is the reachable set of x .

B.2 A NECESSARY CONDITION

Proposition 5 (Ruled variety criterion, Rouchon, 1995) *Suppose the dynamics $\dot{x} = f(x, u)$ is flat. The projection of the sub variety $p = f(x, u)$ in the (p, u) -space (x is here a parameter) onto the p -space is a ruled variety for all x .*

This criterion means that the elimination of u from the n equations $\dot{x} = f(x, u)$ yields $n - m$ equations $F(x, \dot{x}) = 0$ with the following property: for all (x, p) such that $F(x, p) = 0$, there exists $a \in \mathbb{R}^n$, $a \neq 0$ such that

$$\forall \lambda \in \mathbb{R}, \quad F(x, p + \lambda a) = 0$$

The variety $F(x, p)$ is thus ruled since it contains the line passing through p with direction a .

B.3 STATIC STATE FEEDBACK LINEARIZABILITY CRITERION

Consider an affine input system

$$\dot{x} = f(x) + \sum_{i=1}^{n-1} g_i(x) u_i = f(x) + g(x) u$$

where f, g_i are smooth vector fields on a domain $D \subset \mathbb{R}^n$, $x \in D$, $u \in \mathbb{R}^m$.

B.3.1 Brief recall of differential geometry notions

Definition 5 *Let $r \geq 0$ be an integer. A C^r vector field on \mathbb{R}^n is a mapping $f : D \rightarrow \mathbb{R}^n$ of class C^r from an open set $D \subset \mathbb{R}^n$ to \mathbb{R}^n . A smooth vector field is a mapping $f : D \rightarrow \mathbb{R}^n$ of class C^∞ .*

Let $h(x)$ be a smooth vector field on a domain $D \subset \mathbb{R}^n$. The Lie derivative of h along f , denoted as $L_f h(x)$ can be defined (in local coordinates) as

$$\frac{\partial h(x)}{\partial x} f(x) = \sum_{i=1}^n \frac{\partial h(x)}{\partial x_i} f_i(x)$$

since it is a smooth vector field, a Lie derivative operator can be applied to it. Set

$$L_f^i = L_f L_f^{i-1}$$

The *Lie bracket* of f and g can be defined (in local coordinates) as

$$[f, g](x) = \frac{\partial g}{\partial x} f(x) - \frac{\partial f}{\partial x} g(x)$$

Iterated lie brackets are denoted as $\text{ad}_f^i g$:

$$\begin{aligned} \text{ad}_f g &= [f, g] \\ \text{ad}_f^i g &= [f, \text{ad}_f^{i-1} g] \end{aligned}$$

Let f_1, \dots, f_η be some vector fields on $D \subset \mathbb{R}^n$. The *distribution* Δ spanned the vector fields f_1, \dots, f_η is the collection of vector spaces

$$\Delta(x) = \text{span}_{\mathbb{R}^n} \{f_1(x), f_2(x), \dots, f_\eta(x)\}$$

for all $x \in D$. We denote

$$\Delta = \text{span}_{\mathbb{R}^n} \{f_1, f_2, \dots, f_\eta\}$$

A distribution Δ is *involutive* if

$$\forall g_1, g_2 \in \Delta, [g_1, g_2] \in \Delta$$

Proposition 6 *The system with dynamics $\dot{x} = f(x) + g(x)u$ is static state feedback linearizable if, and only if, there is a domain $D_0 \subset D$ such that the following two conditions are satisfied:*

1. *The matrix $[g, \text{ad}_f g, \dots, \text{ad}_f^{n-1} g]$ has rank n for all $x \in D_0$.*
2. *The distribution $\{g, \text{ad}_f g, \dots, \text{ad}_f^{n-2} g\}$ is involutive in D_0 .*



PRECISE DEFINITIONS FOR EXTENSIONS OF DIFFERENTIAL FLATNESS

C.1 SYSTEMS, DYNAMICS AND DIFFERENTIAL FLATNESS

C.1.1 Basic definitions from differential algebra

Definition 6 An ordinary differential field k , is a field on which a mapping $d: k \rightarrow k$ is defined, satisfying the natural properties with respect to addition and product, i.e., for any $x, z \in k$,

$$\begin{aligned}d(x + z) &= d(x) + d(z) \\d(xz) &= d(x)z + xd(z)\end{aligned}$$

Definition 7 Let K be a field. A subfield of K is a subset k of K that is closed under the field operations of K and under taking inverses in K . In other words, k is a field with respect to the field operations inherited from K . The larger field K is then said to be an extension field of k , denoted as K/k .

Definition 8 Let k and K be differential fields with differential operators d_k and d_K respectively. Then, K is a differential extension field of k if K is an extension field of k and

$$\forall x \in k, d_K(x) = d_k(x)$$

Let S be a subset of K . We shall denote by $k\langle S \rangle$ the differential subfield of K generated by k and S .

C.1.2 Algebraic and transcendental extensions

All fields are assumed to be of characteristic zero. Assume also that the differential field extension K/k is *finitely generated*, i.e., there exists a finite subset $S \subset K$ such that $K = k\langle S \rangle$.

Definitions 1 An element a of K is said to be differentially algebraic over k if it satisfies an algebraic differential equation with coefficients in k : there exists a non-zero polynomial P over k , in several indeterminates, such that

$$P(a, \dot{a}, \dots, a^{(v)}) = 0$$

It is said to be differentially transcendental over k if it is not differentially algebraic.

The extension K/k is said to be differentially algebraic if any element of K is differentially algebraic over k . An extension which is not differentially algebraic is said to be differentially transcendental.

C.1.3 Nonlinear systems and flatness

Definition 9 Let k be a given differential ground field. A (nonlinear) system is a finitely generated differential extension K/k .

Definition 10 A nonlinear system K/k is called differentially flat if there exists a finite family $\mathbf{y} = (y_1, \dots, y_m)$ of elements of an algebraic extension L of K such that the extension $L/k\langle\mathbf{y}\rangle$ is (non differentially) algebraic. Such a family is called a flat output.

C.2 H-FIELDS, LIOUVILLIAN AND EXISTENTIAL CLOSEDNESS

The paper (Aschenbrenner et al., 2013) reviews some of the most interesting notions for extending the differential flatness notion. One can also see (Aschenbrenner and van den Dries, 2005a,b, van der Hoeven, 2006) for related material.

Definition 11 An H -field is an ordered differential field K whose natural dominance relation \preccurlyeq satisfies the following two conditions, for all $z \in K$:

(H1) If $z \succ 1$, then $\dot{z}/z > 0$

(H2) If $z \preccurlyeq 1$, then $z - c \prec 1$, for some $c \in C$, the field of constants of K .

Remark 3 In more usual terms, the dominance relations can be explicited as follows, for real valued functions:

$$\circ f \preccurlyeq g \Leftrightarrow \lim_{t \rightarrow \infty} \frac{f(t)}{g(t)} \in \mathbb{R}$$

$$\circ f \prec g \Leftrightarrow \lim_{t \rightarrow \infty} \frac{f(t)}{g(t)} = 0$$

Definition 12 *An H-field K is Liouville closed if it is real closed and any equation $\dot{z} + az = b$ with $a, b \in K$ has a non zero solution in K .*

Definition 13 *An H-field K is existentially closed if every finite system of algebraic differential equations over K in several unknowns with a solution in an H-field extension of K has a solution in K .*

D

TWO LINK ARM INVERSE KINEMATICS

The position of the end effector (the wrist) of a two link arm is given by

$$h_x = l_1 \cos \theta_1 + l_2 \cos(\theta_1 + \theta_2) \quad (\text{D.1a})$$

$$h_y = l_1 \sin \theta_1 + l_2 \sin(\theta_1 + \theta_2) \quad (\text{D.1b})$$

where h_x, h_y are the coordinates of the end effector. By summing the square of the two preceding equations, one obtains

$$\begin{aligned} h_x^2 + h_y^2 &= l_1^2 + l_2^2 + 2l_1l_2 [\cos \theta_1 \cos(\theta_1 + \theta_2) + \sin \theta_1 \sin(\theta_1 + \theta_2)] \\ &= l_1^2 + l_2^2 + 2l_1l_2 \cos \theta_2 \end{aligned}$$

Then

$$\cos \theta_2 = \frac{h_x^2 + h_y^2 - l_1^2 - l_2^2}{2l_1l_2}$$

or

$$\begin{aligned} \theta_2 &= \arctan \left(\frac{\sin \theta_2}{\cos \theta_2} \right) \\ &= \arctan \left(\frac{\pm \sqrt{1 - \cos^2 \theta_2}}{\cos \theta_2} \right) \\ &= \arctan \left(\pm \frac{\sqrt{1 - \bar{h}^2}}{\bar{h}} \right) \end{aligned}$$

$$\bar{h} = \frac{h_x^2 + h_y^2 - l_1^2 - l_2^2}{2l_1l_2}$$

Setting

$$k_1 = l_1 + l_2 \cos \theta_2, \quad k_2 = l_2 \sin \theta_2$$

one has

$$\begin{aligned}h_x &= k_1 \cos \theta_1 - k_2 \sin \theta_1 \\h_y &= k_1 \sin \theta_1 + k_2 \cos \theta_1\end{aligned}$$

Then

$$\rho = \sqrt{k_1^2 + k_2^2}, \quad \gamma = \arctan \left(\frac{k_2}{k_1} \right)$$

wherefrom

$$\begin{aligned}h_x &= \rho \cos \gamma \cos \theta_1 - \rho \sin \gamma \sin \theta_1 \\h_y &= \rho \cos \gamma \sin \theta_1 + \rho \sin \gamma \cos \theta_1\end{aligned}$$

or, what is the same

$$\frac{h_x}{\rho} = \cos(\gamma + \theta_1), \quad \frac{h_y}{\rho} = \sin(\gamma + \theta_1)$$

Then

$$\theta_1 + \gamma = \arctan \left(\frac{h_y}{h_x} \right)$$

and, finally

$$\theta_1 = \arctan \left(\frac{h_y}{h_x} \right) - \arctan \left(\frac{l_2 \sin \theta_2}{l_1 + l_2 \cos \theta_2} \right)$$

E

TWO LINK ARM CODE LISTING

E.1 FLATNESS BASED CONTROL OF THE ARM

The listing beginning on the next page is a matlab code of the differential flatness based control of the two link arm example whose model is depicted in (3.13) and tracking feedback law in (3.25).

```

1 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
  % Simulation of a two joint arm flatness based tracking
  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
  function mainTwoJointArmFlatness()
      %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
6      % Nomenclature
      % P: physical parameters
      % R: reference trajectories and control laws
      % U: complete control laws
      % G: gains
11     % S: simulation scenario
      %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

      clear all
      %
16     %% PHYSICAL parameters
      % Fixed physical parameters
      P.g = 9.81;          % gravity constant
      P.l1 = 0.3384;       % (m) length of lower arm part
      P.l2 = 0.4554;       % (m) length of upper arm part
21     P.r1 = 0.1692;       % (m) distance to center of mass
      P.r2 = 0.2277;       % (m) distance to center of mass
      P.m1 = 2.10;         % (kg) mass of lower arm part
      P.m2 = 1.65;         % (kg) mass of upper arm part
      P.J1 = 0.025;        % (kg.m^2) inertia of lower arm part
26     P.J2 = 0.075;        % (kg.m^2) inertia of upper arm part

      %% REFERENCE trajectory
      % A tanh one
      R.hxi = 0.8*(P.l1+P.l2); R.hxf = 0;
31     R.hyi = P.l1 + 0.1*P.l2; R.hyf = - 0.1*P.l1;
      R.stiffness = 9; R.hxr = 0.5*R.hxi;

      %% SIMULATION values
      S.tini = 0;          S.tend = 10;          % intial and ←
          final simulation times
36     S.relTol = 1e-3;    S.absTol = 1e-6;      % relative and ←
          absolute simul tols
      % Intial state values
      tvirt = [S.tini:0.01:S.tend]';
      strRef = inverseKinematics(tvirt, P, R, S);
      S.theta10 = strRef.theta1r(1) + 0.05*(strRef.←
          theta1r(end)-strRef.theta1r(1));
41     S.dotTheta10 = strRef.dotTheta1r(1) + 0.05*(strRef.←
          dotTheta1r(end)-strRef.dotTheta1r(1));
      S.theta20 = strRef.theta2r(1) + 0.05*(strRef.←
          theta1r(end)-strRef.theta1r(1));
      S.dotTheta20 = strRef.dotTheta2r(1) + 0.05*(strRef.←
          dotTheta2r(end)-strRef.dotTheta2r(1));

```

```

%% Default FEEDBACK GAINS
46 % flatness based gains
% for  $(s+s_1)(s+s_2) = s^2 + (s_1+s_2)s + s_1s_2$ 
sTh1 = 5; sDotTh1 = 2*sTh1;
sTh2 = 6; sDotTh2 = 2*sTh2;
K.kp1 = sTh1*sDotTh1; K.kp2 = sTh2*sDotTh2;
51 K.kd1 = sTh1+sDotTh1; K.kd2 = sTh2+sDotTh2;
K.Kp = [K.kp1 0; 0 K.kp2];
K.Kd = [K.kd1 0; 0 K.kd2];

% Saving option
56 S.forSaving = 'yes';

% Simulate
options = odeset('RelTol', S.relTol, 'AbsTol', S.↵
    absTol);
[ts Xs] = ode23tb(@dynTwoJointArm, [S.tini S.tend], ↵
    ...
61 [S.theta10 S.dotTheta10 S.theta20 S.↵
    dotTheta20]',...
    options, P, R, K, S);
S.ts = ts; S.Xs = Xs;
plotVariables(P, R, K, S);

66 end % of main()

function P = coriolisGravity(P, theta1, dotTheta1, theta2↵
, dotTheta2)
% Masses and inertia gathering
71 m1 = P.m1; m2 = P.m2; l1 = P.l1; l2 = P.l2;
r1 = P.r1; r2 = P.r2; J1 = P.J1; J2 = P.J2;
% Coriolis and gravity terms computations
M11 = J1 + J2 + (m1*r1^2) + (m2*((l1^2) + (r2^2) + ↵
    (2*l1*r2*cos(theta2))));
M12 = J2 + (m2*((r2^2) + (l1*r2*cos(theta2))));
76 M21 = M12;
M22 = J2 + (m2*r2^2);
M = [M11,M12; M21,M22];
C1 = -(m2*l1*r2*dotTheta2^2*sin(theta2)) -...
    (2*m2*l1*r2*dotTheta1*dotTheta2*sin(theta2));
81 C2 = m2*l1*dotTheta1^2*r2*sin(theta2);
C = [C1, C2]';
G1 = (P.g*sin(theta1)*((m2*l1)+(m1*r1))) + (P.g*m2*↵
    r2*sin(theta1+theta2));
G2 = P.g*m2*r2*sin(theta1+theta2);
G = [G1, G2]';
86 P.C = C; P.G = G; P.M = M;

end

```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Dynamics
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
91 function [dotX] = dynTwoJointArm(t, X, P, R, K, S)
    persistent count;

    if (t <= 0) count = 0; end;
96    if (round(t) >= count)
        count = count + 1;
        disp(sprintf('time t = %f\n', t));
    end;
    % Variable gathering
101    dotX = zeros(4,1);
    theta1 = X(1,:); dotTheta1 = X(2,:);
    theta2 = X(3,:); dotTheta2 = X(4,:);
    % Coriolis and gravity terms computations
    P = coriolisGravity(P, theta1, dotTheta1, theta2, dotTheta2);
106    % Control law computation
    T = twoLinkFlatCtrlLaw(t, X, P, R, K, S);
    % Two link arm DYNAMICS
    M = P.M; C = P.C; G = P.G;
    ddotTheta = inv(M) * (-C - G + T);
111    % return the derivative of the state
    dotX = [dotTheta1 ddotTheta(1) dotTheta2 ddotTheta(2)]';

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
116 % Flatness based tracking control law
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [T] = twoLinkFlatCtrlLaw(t, X, P, R, K, S)
    % Variable gathering
    dotX = zeros(4,1);
121    theta1 = X(1,:); dotTheta1 = X(2,:);
    theta2 = X(3,:); dotTheta2 = X(4,:);
    M = P.M; C = P.C; G = P.G; l1 = P.l1; l2 = P.l2;
    % reference trajectories
    [strRefHx strRefHy] = tanhRefTraj(t, P, R, S);
126    hxr = strRefHx.v; hyr = strRefHy.v;
    dotHxr = strRefHx.d1; dotHyr = strRefHy.d1;
    ddotHxr = strRefHx.d2; ddotHyr = strRefHy.d2;
    % Intermediary computations
    [Hinv phi] = computeHinvPhi(theta1, theta2, dotTheta1, dotTheta2, P);
131    hx = l1*cos(theta1) + l2*cos(theta1+theta2);
    hy = l1*sin(theta1) + l2*sin(theta1+theta2);
    dotHx = -l1*dotTheta1*sin(theta1) - l2*(dotTheta1+dotTheta2)*sin(theta1+theta2);

```

```

dotHy = l1*dotTheta1*cos(theta1) + l2*(dotTheta1+↵
    dotTheta2)*cos(theta1+theta2);
% errors computation
136 ehx = hx - hxr;      ehy = hy - hyr;
dotEHx = dotHx - dotHxr; dotEHy = dotHy - dotHyr;
% Vector computation
vectDdotHr = [ddotHxr ddotHyr]';
vectEH = [ehx ehy]';
141 vectDotEH = [dotEHx dotEHy]';
% control law computation
T = C + G + M*Hinv*(phi + vectDdotHr -...
    K.Kd * vectDotEH - K.Kp * ↵
    vectEH);

end

146 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Compute phi and the inverse of H
% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [matHinv vectPhi] = computeHinvPhi(theta1, ↵
    theta2, dotTheta1, dotTheta2, P)
151 l1 = P.l1; l2 = P.l2;
hx = l1*cos(theta1) + l2*cos(theta1+theta2);
hy = l1*sin(theta1) + l2*sin(theta1+theta2);
H = [-hy -l2*sin(theta1+theta2);
    hx l2*cos(theta1+theta2)];
156 % matHinv = [-l2*cos(theta1+theta2) -l2*sin(theta1+↵
    theta2);
%
% hx hy ↵
%
% ]...
%
% / (l2*(hy*cos(theta1+theta2)-hx*sin(↵
    theta1+theta2)));
matHinv = inv(H);
vectPhi = [l1*dotTheta1^2*cos(theta1) + l2*(↵
    dotTheta1+dotTheta2)^2*cos(theta1+theta2);
161 l1*dotTheta1^2*sin(theta1) + l2*(↵
    dotTheta1+dotTheta2)^2*sin(theta1+↵
    theta2)];

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Reference trajectory
166 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [strRefHx strRefHy] = tanhRefTraj(t, P, R, S)
hx = ((R.hxf - R.hxi).*t )./(S.tend) + R.hxi;
strRefHx.v = hx;
strRefHx.d1 = ((R.hxf - R.hxi)./S.tend).*ones(length(hx)↵
    ),1);
171 strRefHx.d2 = zeros(length(hx),1);
strRefTanh = tanhTr(hx, R.stiffness, R.hyi, R.hyf, R.↵
    hxR);

```

```

    strRefHy.v = strRefTanh.v;
    strRefHy.d1 = strRefTanh.d1 .* strRefHx.d1;
    strRefHy.d2 = strRefTanh.d2 .* strRefHx.d1.^2;
176 end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Inverse kinematics
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
181 function strRef = inverseKinematics(t, P, R, S)
    l1 = P.l1; l2 = P.l2;
    [strRefHx strRefHy] = tanhRefTraj(t, P, R, S);
    hxr = strRefHx.v;    hyr = strRefHy.v;
    dothxr = strRefHx.d1; dothyr = strRefHy.d1;
186 ddothxr = strRefHx.d2; ddothyr = strRefHy.d2;
    cosTheta2r = (hxr.^2 + hyr.^2 - l1.^2 - l2.^2) ./ (←
        (2.*l1.*l2));
    sinTheta2r = sqrt(1 - cosTheta2r.^2);
    theta2r = unwrap(atan2(sqrt(1 - cosTheta2r.^2),←
        cosTheta2r));
    theta1r = unwrap(atan2(hyr,hxr)) -...
191         unwrap(atan2(l2.*sinTheta2r,(l1+l2.*←
            cosTheta2r)));
    dotTheta2r = -(hxr.*dothxr + hyr.*dothyr) ./ (l1*l2←
        .*sinTheta2r);
    dotTheta1r = (dothyr.*hxr - hyr.*dothxr) ./ (hxr.^2 ←
        + hyr.^2) -...
        (l2.*dotTheta2r.*(1+l1.*cosTheta2r)) ←
        ./...
        (l1^2 + 2*l1*l2.*cosTheta2r + l2^2);
196 ddotTheta2r = -(hxr.*ddothxr + dothxr.^2 + hyr.*←
        ddothyr + dothyr.^2) ./...
        (l1*l2.*sinTheta2r) + (hxr.*dothxr + ←
        hyr.*dothyr) .*...
        (cosTheta2r.*dotTheta2r) ./ (l1*l2.*←
        sinTheta2r.^2);
    ddotTheta1r = (ddothyr.*hxr - hyr.*ddothxr) ./ (hxr←
        .^2 + hyr.^2) -...
        2.*(dothyr.*hxr - hyr.*dothxr).*(hxr.*←
        dothxr + hyr.*dothyr) -...
201        12.*(ddotTheta2r.*(1+l1.*cosTheta2r) -←
        (l1.*dotTheta2r.^2.*sinTheta2r)) ←
        ./...
        (l1^2 + 2*l1*l2.*cosTheta2r + l2^2) + ←
        2*l1*l2.*(l2.*dotTheta2r.*(1+l1 ←
        .*...
        cosTheta2r)).*(sinTheta2r.*dotTheta2r)←
        ./ (l1^2+2*l1*l2.*cosTheta2r+l2^2)←
        ^2).^2;
    strRef.theta1r = theta1r;    strRef.theta2r ←
        = theta2r;

```

```

    strRef.dotTheta1r = dotTheta1r; strRef.dotTheta2r ←
        = dotTheta2r;
206 strRef.ddotTheta1r = ddotTheta1r; strRef.ddotTheta2r ←
        = ddotTheta2r;

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% plots
211 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function plotVariables(P, R, K, S)
%
% For plots
police = 'Helvetica'; size = 24; lineWidth = 2;

216 % reference state
ts = S.ts; tr = ts; Xs = S.Xs;
% Reference curve and inverse kinematics
[strRefHx strRefHy] = tanhRefTraj(tr, P, R, S);
221 hxr = strRefHx.v; hyr = strRefHy.v;
dotHxr = strRefHx.d1; dotHyr = strRefHy.d1;
ddotHxr = strRefHx.d2; ddotHyr = strRefHy.d2;
strRefTh = inverseKinematics(tr, P, R, S);
theta1r = strRefTh.theta1r; theta2r = ←
    strRefTh.theta2r;
226 dotTheta1r = strRefTh.dotTheta1r; dotTheta2r = ←
    strRefTh.dotTheta2r;

% simulated state
theta1 = Xs(:,1); dotTheta1 = Xs(:,2);
theta2 = Xs(:,3); dotTheta2 = Xs(:,4);
% Control laws computation
231 T1r = []; T2r = []; T1 = []; T2 = [];
XsT = Xs';
for i = 1:length(ts)
    % open loop control
    Pr = coriolisGravity(P, theta1r(i), dotTheta1r(i)←
        , theta2r(i), dotTheta2r(i));
236 Mr = Pr.M; Cr = Pr.C; Gr = Pr.G;
[Hinvr phir] = computeHinvPhi(theta1r(i), theta2r←
    (i), dotTheta1r(i), dotTheta2r(i), P);
DdotHr = [ddotHxr(i) ddotHyr(i)]';
Tr = Cr + Gr - Mr*Hinvr*(DdotHr + phir);
T1r = [T1r; Tr(1)]; T2r = [T2r; Tr(2)];
241 % simulated closed loop control
P = coriolisGravity(P, theta1(i), dotTheta1(i), ←
    theta2(i), dotTheta2(i));
T = twoLinkFlatCtrlLaw(ts(i), XsT(:,i), P, R, K, ←
    S);
T1 = [T1; T(1)]; T2 = [T2; T(2)];
end;
246 % Simulated curve: forward kinematics

```

```

11 = P.11;    12 = P.12;
hx          = 11.*cos(theta1) + 12 .* cos(theta1 + theta2)↵
;
hy          = 11.*sin(theta1) + 12 .* sin(theta1 + theta2)↵
;

251  figure(1);
      % hx hy plot
      subplot(2, 2, 1);
      plot(tr, hxr, 'r', ts, hx, 'b', 'LineWidth', ↵
            lineWidth); grid;
      xlabel('time (s)', 'FontName', police, 'FontSize', ↵
            size);
256  ylabel('h_x, h_{xr} (deg)', 'FontName', police, '↵
            FontSize', size);
      title('red h_{xr} ; blue h_x',...
            'FontName', police, 'FontSize', size, '↵
            FontWeight','bold');

      % theta2 plot
      subplot(2, 2, 2);
261  plot(tr, hyr, 'r', ts, hy, 'b', 'LineWidth', ↵
            lineWidth); grid;
      xlabel('time (s)', 'FontName', police, 'FontSize', ↵
            size);
      ylabel('h_y, h_{yr} (deg)', 'FontName', police, '↵
            FontSize', size);
      title('red h_{yr} ; blue h_y',...
            'FontName', police, 'FontSize', size, '↵
            FontWeight','bold');

266  subplot(2, 2, 3);
      plot(tr, theta1r.*(180/pi), 'r', ts, theta1.*(180/pi)↵
            , 'b', 'LineWidth', lineWidth); grid;
      xlabel('time (s)', 'FontName', police, 'FontSize', ↵
            size);
      ylabel('\theta_{2}, \theta_{2r} (deg)', 'FontName', ↵
            police, 'FontSize', size);
      title('red \theta_{1r} ; blue \theta_1',...
            'FontName', police, 'FontSize', size, '↵
271  FontWeight','bold');

      subplot(2, 2, 4);
      plot(tr, theta2r.*(180/pi), 'r', ts, theta2.*(180/pi)↵
            , 'b', 'LineWidth', lineWidth); grid;
      xlabel('time (s)', 'FontName', police, 'FontSize', ↵
            size);
      ylabel('\theta_{2}, \theta_{2r} (deg)', 'FontName', ↵
            police, 'FontSize', size);
276  title('red \theta_{2r} ; blue \theta_2',...
            'FontName', police, 'FontSize', size, '↵
            FontWeight','bold');

```



```

hFig2 = figure(2);
% T1 and T1r plot
281 set(gca, 'FontName', police, 'FontSize', size);
plot(tr, T1r, 'r--', ts, T1, 'b-', 'LineWidth', ←
    lineWidth); grid;
xlabel('time (s)', 'FontName', police, 'FontSize', ←
    size);
ylabel('T_1, T_{1r} (N)', 'FontName', police, '←
    FontSize', size);
if (strcmp(S.forSaving, 'yes') ~= 1)
286     title('red T_{1r} ; blue T_1',...
        'FontName', police, 'FontSize', size, '←
        FontWeight', 'bold');
else
    print(hFig2, '-dpdf', '../GraphicsImages/←
        twoLinkArmCtrl1.pdf');
end

291 hFig3 = figure(3);
% T2 and T2r plot
set(gca, 'FontName', police, 'FontSize', size);
plot(tr, T2r, 'r--', ts, T2, 'b-', 'LineWidth', ←
    lineWidth); grid;
296 xlabel('time (s)', 'FontName', police, 'FontSize', ←
    size);
ylabel('T_2, T_{2r} (N)', 'FontName', police, '←
    FontSize', size);
if (strcmp(S.forSaving, 'yes') ~= 1)
    title('red T_{2r} ; blue T_2',...
        'FontName', police, 'FontSize', size, '←
        FontWeight', 'bold');
301 else
    print(hFig3, '-dpdf', '../GraphicsImages/←
        twoLinkArmCtrl2.pdf');
end

hFig4 = figure(4);
306 set(gca, 'FontName', police, 'FontSize', size);
plot(tr, hx-hxr, 'b', 'LineWidth', lineWidth); grid;
xlabel('time (s)', 'FontName', police, 'FontSize', ←
    size);
ylabel('h_x - h_{xr} (m)', 'FontName', police, '←
    FontSize', size);
if (strcmp(S.forSaving, 'yes') ~= 1)
311     title('h_x - h_{xr}',...
        'FontName', police, 'FontSize', size, '←
        FontWeight', 'bold');
else
    print(hFig4, '-dpdf', '../GraphicsImages/←
        twoLinkArmErrHx.pdf');

```

```

end
316 hFig5 = figure(5);
set(gca, 'FontName', police, 'FontSize', size);
plot(tr, hy-hyr, 'b', 'LineWidth', lineWidth); grid;
xlabel('time (s)', 'FontName', police, 'FontSize', size);
ylabel('h_y - h_{yr} (m)', 'FontName', police, 'FontSize', size);
321 if (strcmp(S.forSaving, 'yes') ~= 1)
title('h_y - h_{yr}',...
'FontName', police, 'FontSize', size, 'FontWeight','bold');
else
326 print(hFig5, '-dpdf', '../GraphicsImages/twoLinkArmErrHy.pdf');
end

% forward reference and actual kinematics
hFig6 = figure(6);
331 set(gca, 'FontName', police, 'FontSize', size);
plot(hx, hy, 'b-', hxr, hyr, 'r--', 'LineWidth', lineWidth); grid;
xlabel('h_x (m)', 'FontName', police, 'FontSize', size);
ylabel('h_y (m)', 'FontName', police, 'FontSize', size);
if (strcmp(S.forSaving, 'yes') ~= 1)
336 title('blue h_x h_y red h_{xr} h_{yr}',...
'FontName', police, 'FontSize', size, 'FontWeight','bold');
else
print(hFig6, '-dpdf', '../GraphicsImages/twoLinkArmHxHy.pdf');
end

341 hFig7 = figure(7);
set(gca, 'FontName', police, 'FontSize', size);
R.hxi = 0.8*(P.l1+P.l2); R.hxf = 0;
R.hyi = P.l1 + 0.1*P.l2; R.hyf = - 0.1*P.l1;
346 plot([], []); grid;
hold on
for i = 1:3:length(ts)
line([0, l1*cos(theta1(i))], [0, l1*sin(theta1(i))], 'Color', [0 0 0.5], 'LineWidth', 2);
line([l1*cos(theta1(i)), l1*cos(theta1(i))+l2*cos(theta1(i)+theta2(i))],...
351 [l1*sin(theta1(i)), l1*sin(theta1(i))+l2*sin(theta1(i)+theta2(i))],...

```

```

        l1*sin(theta1(i))+l2*sin(theta1(i)+theta2(i)↵
        )]], 'Color', [0 0 0.9], 'LineWidth', ↵
        2);
    plot(hx(i), hy(i), 'ro', 'LineWidth', 2); grid;
    pause(0.01);
end
356 hold off
xlabel('h_x (m)', 'FontName', police, 'FontSize', ↵
    size);
ylabel('h_y (m)', 'FontName', police, 'FontSize', ↵
    size);
if (strcmp(S.forSaving, 'yes') ~= 1)
    title('blue h_x h_y  red h_{xr} h_{yr}',...
361     'FontName', police, 'FontSize', size, '↵
        FontWeight','bold');
else
    print(hFig7, '-dpdf', '../GraphicsImages/↵
        twoLinkArmAnimation.pdf');
end
366 end

```
